RM0449
Reference manual

SPC58 4B Line - 32 bit Power Architecture automotive MCU z4 core 120 MHz, 2 MBytes Flash, HSM, ASIL-B

Overview

The SPC584Bx is a family of Power Architecture based microcontrollers that targets automotive vehicle body and gateway applications such as:

- Central body controller
- Smart junction box
- Mid and high end gateway

The SPC584Bx belong to a wide family of automotive microcontroller products which offers the scalability needed to implement platform approaches and delivers the performance and features required by increasingly sophisticated body applications.

It is available as single or symmetrical dual core and operates at speeds of up to 120 MHz enabling the customer to adjust the performance and consumption to the application needs. The versatile low power modes available on SPC56 body MCU have been extended with a Smart Standby Wake-up Unit reducing further the average consumption in low power conditions.
A large variety and number of communication interfaces like ISO CAN-FD, Ethernet with AVB are available as well as new features for security (HSM) and safety (ASIL-B) requirements.

It also capitalizes on the nominal available development infrastructure of current Power Architecture devices and is supported with software drivers, operating systems and configuration code to assist with users implementations.

The information in this book is subject to change without notice, as described in the disclaimer. As with any technical documentation, it is the readers' responsibility to ensure they have the most current version of this document.

To locate any published errata or updates for this document, visit the ST Web site at http://www.st.com.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the SPC584Bx device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

Contents

1 Preface

1.1 Document organization

This document includes chapters that are divided into parts:

- Part I includes chapters that describe the device as a whole or provide device-specific information.
- Part II includes chapters that describe the functionality of the individual modules on the device.

1.2 Register conventions

Figure 1 and *Table 1* provide the register conventions used throughout this manual.

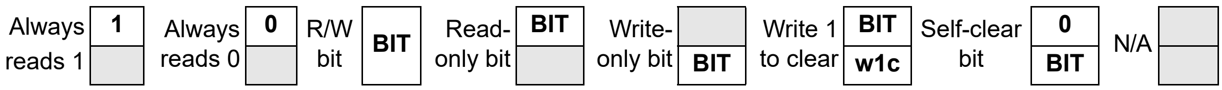


Figure 1. Key to register fields

Table 1. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register field types	
R	Read only. Writing this bit has no effect.
W	Write only
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset values	
0	Resets to zero
1	Resets to one
—	Undefined at reset

1.3 Acronyms and abbreviations

Appendix A: Acronyms and abbreviations lists acronyms and abbreviations used in this document.

1.4 Reference documents

In addition to this reference manual, the following documents provide additional information on the operation of the SPC584Bx:

- IEEE-ISTO 5001™-2012 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture

Information about Power ISA™ Book is available at Power Architecture web site.

2 Introduction

2.1 SPC584Bx microcontroller

The SPC584Bx microcontroller is a new device in the SPC58 family superseding the SPC564Cx and SPC56ECx family. SPC584Bx is built on the legacy of the SPC564Cx and SPC56ECx family, while introducing new features coupled with higher throughput to provide substantial reduction of cost per feature and significant power and performance improvement (MIPS per mW). On the SPC584Bx device, there is one processor core e200z420 and one e200z0 core embedded in the Hardware Security Module.

2.1.1 Core features

The platform consists in one e200z420 CPU operating at up to 120 MHz. It includes enhancements to implement safety (e2e ECC, Core MPU) and increased performance (Local D-RAM as well as Instruction and Data Cache).

The Hardware Security Module (HSM) includes one CPU running at 60 MHz.

All CPUs are compatible with the Power Architecture VLE instruction set, that supports code size reduction. The Power Architecture has enhancements that improve the architecture's fit in embedded applications.

2.1.2 Memory hierarchy

The SPC584Bx has three levels of memory hierarchy. At the top level there is a combination of Instruction Cache, Data Cache, and Local Data RAM. The local RAM arrays are embedded into the CPU's pipeline to provide the fastest possible access time. The second level of memory is system RAM and flash. These are connected via a 120 MHz, 64-bit wide crossbar. The third level of memory system is the I/O subsystem. All the I/O peripherals are connected together via a 120 MHz 64-bit wide crossbar switch, while peripherals are connected at different frequencies (see the clocking chapter for further details). The reduced speed for the peripherals helps to reduce power.

2.2 Features

On-chip modules within SPC584Bx include the following features:

- One main CPU, dual issue, 32-bit CPU core complexes (e200z4).
  - Power Architecture embedded specification compliance
  - Instruction set enhancement allowing variable length encoding (VLE), encoding a mix of 16-bit and 32-bit instructions, for code size footprint reduction
  - Single-precision floating point operations
  - 64 KB local data RAM for Core_2
  - 8 KB I-Cache and 4 KB D-Cache for Core_2
- 2112 KB (2048 KB code flash + 64 KB data flash) on-chip flash memory
  - Supports read during program and erase operations, and multiple blocks allowing EEPROM emulation
- 176 KB HSM dedicated flash memory (144 KB code + 32 KB data)
- 128 KB on-chip general-purpose SRAM (+ 64 KB local data RAM: 64 KB included in the CPU)
- Multi channel direct memory access controllers
  - 32 eDMA channels
- One interrupt controller (INTC)
- Dual phase-locked loops with stable clock domain for peripherals and FM modulation domain for computational shell
- Crossbar switch architecture for concurrent access to peripherals, Flash, or RAM from multiple bus masters with end-to-end ECC
- Hardware security module (HSM) with HW cryptographic co-processor
- System integration unit lite (SIUL)
- Boot assist Flash (BAF) supports factory programming using a serial bootload through the asynchronous CAN or LIN/UART.
- Hardware support for safety ASIL-B level related applications
- Enhanced modular IO subsystem (eMIOS): up to 64 (2 x 32) timed I/O channels with 16-bit counter resolution
  - Buffered updates
  - Support for shifted PWM outputs to minimize occurrence of concurrent edges
  - Supports configurable trigger outputs for ADC conversion for synchronization to channel output waveforms
  - Shared or independent time bases
  - DMA transfer support available
- Body cross triggering unit (BCTU)
  - Triggers ADC conversions from any eMIOS channel
  - Triggers ADC conversions from up to 2 dedicated PIT_RTIs
  - One event configuration register dedicated to each timer event allows to define the corresponding ADC channel
  - Synchronization with ADC to avoid collision
- Enhanced analog-to-digital converter system with:
  - Two independent fast 12-bit SAR analog converters

- One supervisor 12-bit SAR analog converter
- One 10-bit SAR analog converter with STDBY mode support
- Seven deserial serial peripheral interface (DSPI) modules
- Fourteen LIN and UART communication interface (LINFlexD) modules
  - LINFlexD_0 is a Master/Slave
  - All others are Masters
- Eight modular controller area network (MCAN) modules, all supporting flexible data rate (ISO CAN-FD compliant)
- One ethernet controller 10/100 Mbps, compliant to IEEE 802.3-2008
  - IEEE 1588-2008 Time stamping (internal 64-bit time stamp)
  - IEEE 802.1AS and IEEE 802.1Qav (AVB-Feature)
  - IEEE 802.1Q VLAN tag detection
  - IPv4 and IPv6 checksum modules
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2010 standard.
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1 and IEEE 1149.7), 2-pin JTAG interface.
- Standby power domain with smart wake-up sequence

2.3 Feature list

Table 2 lists a summary of major features for the SPC584Bx device. The feature column represents a combination of module names and capabilities of certain modules. A detailed description of the functionality provided by each on-chip module is given later in this document.

Table 2. Features List

Feature	Description
SPC58 family	40 nm
Number of Cores	1
Local RAM	64 KB Data
Single Precision Floating Point	Yes
SIMD	No
VLE	Yes
Cache	8 KB Instruction
	4 KB Data
MPU	Core MPU: 24 per CPU
	System MPU: 24 per XBAR
Semaphores	No
CRC Channels	2 x 4
Software Watchdog Timer (SWT)	2

Table 2. Features List (continued)

Feature	Description
Core Nexus Class	3+
Event Processor	4 x SCU
	4 x PMC
Run control Module	Yes
System SRAM	128 KB (full standby RAM)
Flash	2048 KB code / 64 KB data
Flash fetch accelerator	2 x 4 x 256-bit
DMA channels	32
DMA Nexus Class	3
LINFlexD	14
MCAN (ISO CAN-FD compliant)	8
DSPI	7
I2C	1
Ethernet	1 MAC with Time Stamping, AVB and VLAN support
SIPI / LFAST Debugger	High Speed
System Timers	8 PIT channels
	1 AUTOSAR® (STM)
	RTC/API
eMIOS	2 x 32 channels
BCTU	64 channels
ADC (SAR)	4
Temp. sensor	Yes
Self Test Controller	Yes
PLL	Dual PLL with FM
Integrated linear voltage regulator	Yes
External Power Supplies	5 V, 3.3 V
Low Power Modes	HALT Mode
	STOP Mode
	Smart Standby with output controller, analog and digital inputs
	Standby Mode

2.4 Packages

The SPC584Bx is available in these production packages: eTQFP64, eTQFP100, eTQFP144, eLQFP176.

2.5 Software debug and calibration

The Production Device (PD) includes many features to facilitate software development. These features include:

- Nexus code execution trace on all CPUs
- Nexus data trace on all bus masters (DMA, Interprocessor bus, CPUs, etc.)
- Performance monitors for each CPU
- Sequence Processing Unit for complex trace triggering

2.6 Block diagram

The figures below show the top-level block diagrams.

Figure 2. Block diagram

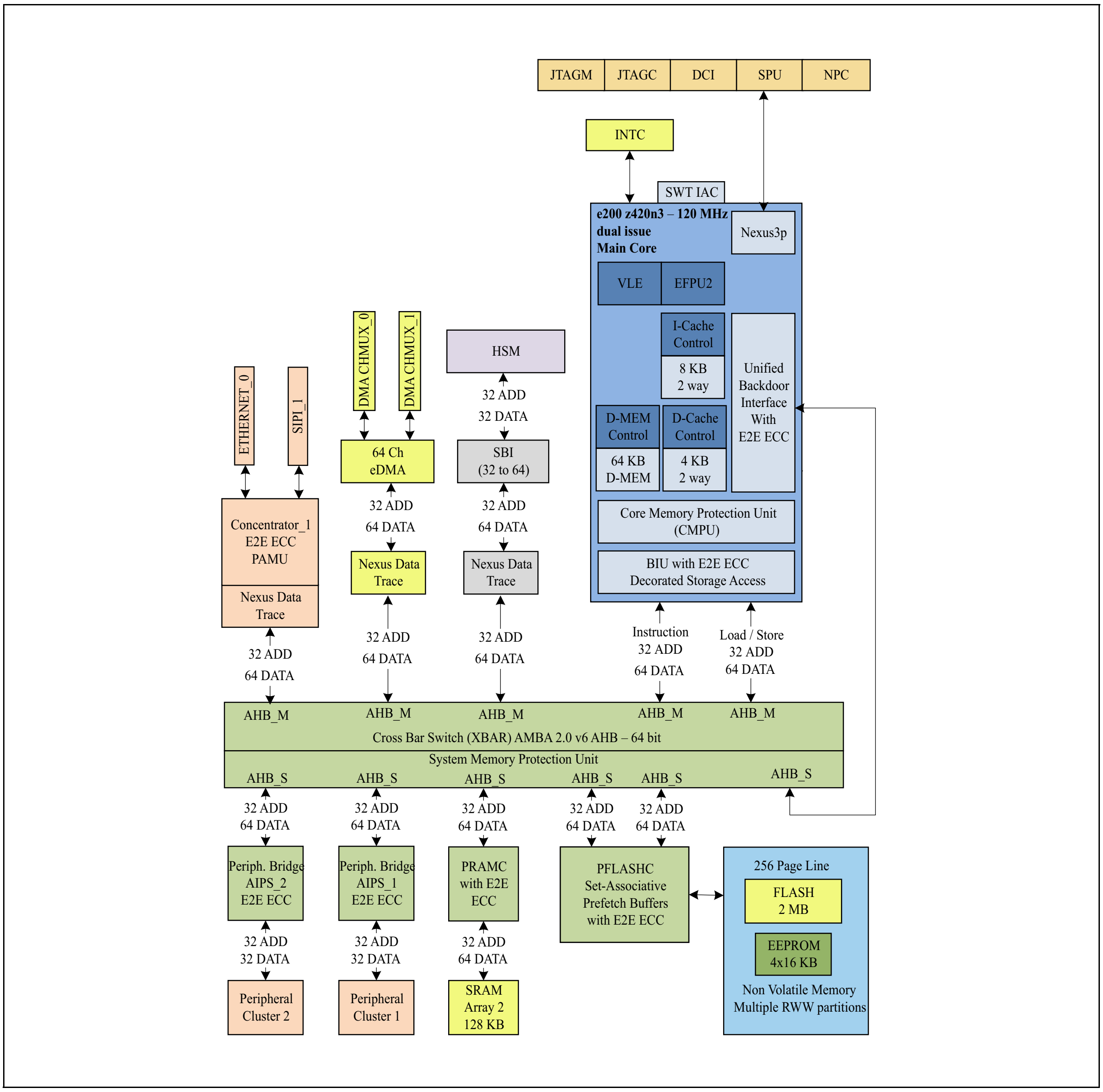
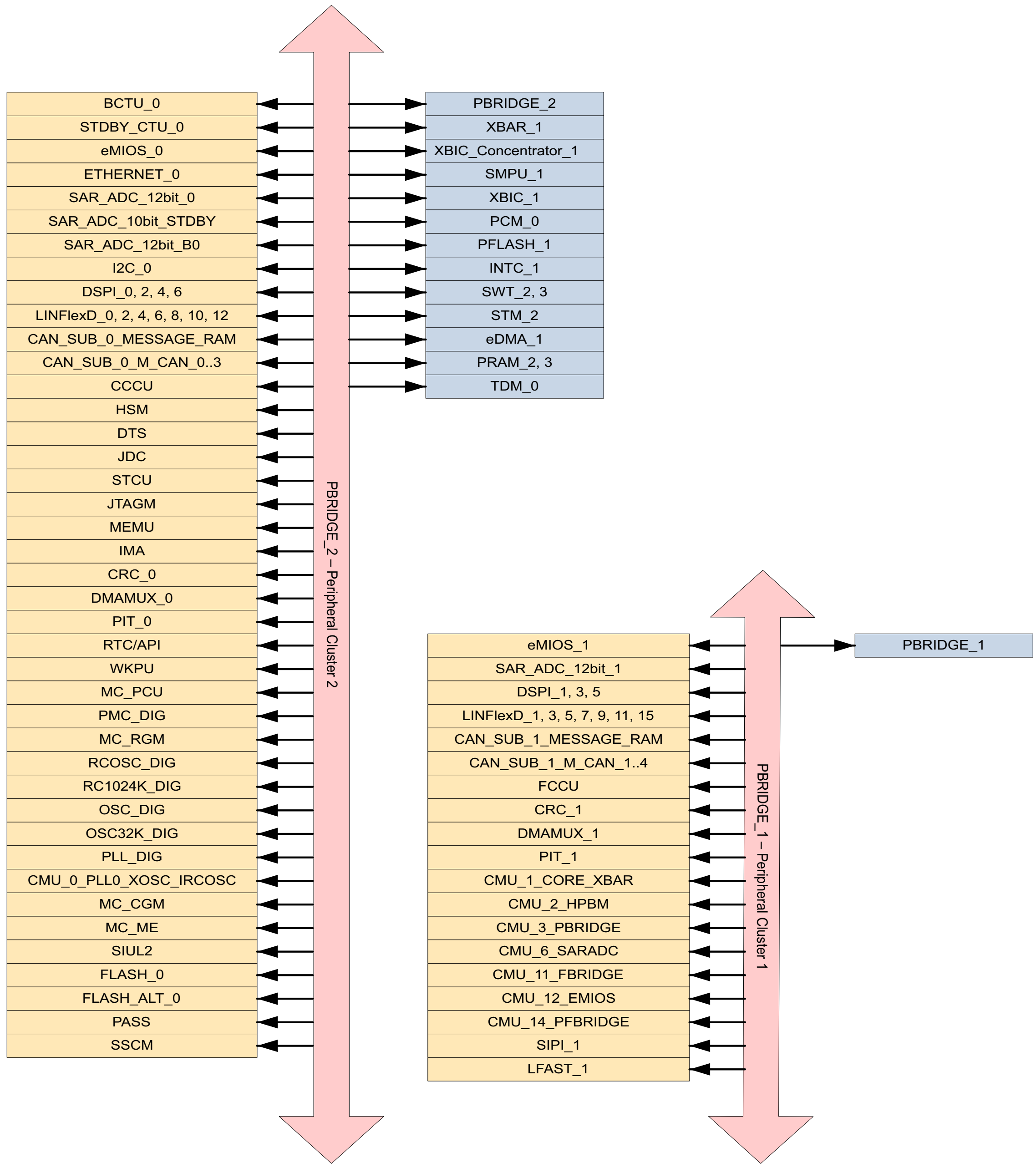


Figure 3. Periphery allocation



Note: In this diagram, ON-platform modules are shown in orange color and OFF-platform modules are shown in blue color.

3 Embedded memories

3.1 Overview

Features include:

- Onboard SRAM, including system SRAM, and data memory for the processor core
- Onboard system error correction code (ECC) Flash memory
- End-to-end ECC (e2eECC) error detection and correction
- Built-in Flash memory security features including censorship and a programmable Hardware Security Module (HSM)
- Embedded memories in the peripherals[a]:
  - CAN
  - eDMA
  - Ethernet MAC

3.2 SRAM

3.2.1 System SRAM

SPC584Bx device includes 128 KB of general-purpose on-chip ECC SRAM, including 128 KB standby RAM. The SRAM can be configured for either zero- or one-wait state read operation latency using the PRCR1 register in the SRAM controller.

- Refer to *Chapter 33: Platform RAM controller AHB (PRAMC_AHB)* for details on configuring the SRAM controller.
- Refer to the Memory Map chapter for the SPC584Bx SRAM map.

3.2.2 Processor core local RAM

SPC584Bx devices include local data RAM (D-MEM) for the processor core to provide enhanced performance.

Refer to the Introduction chapter for processor core local RAM size.

The core RAM supports zero wait-state, single-cycle latency on processor local accesses.

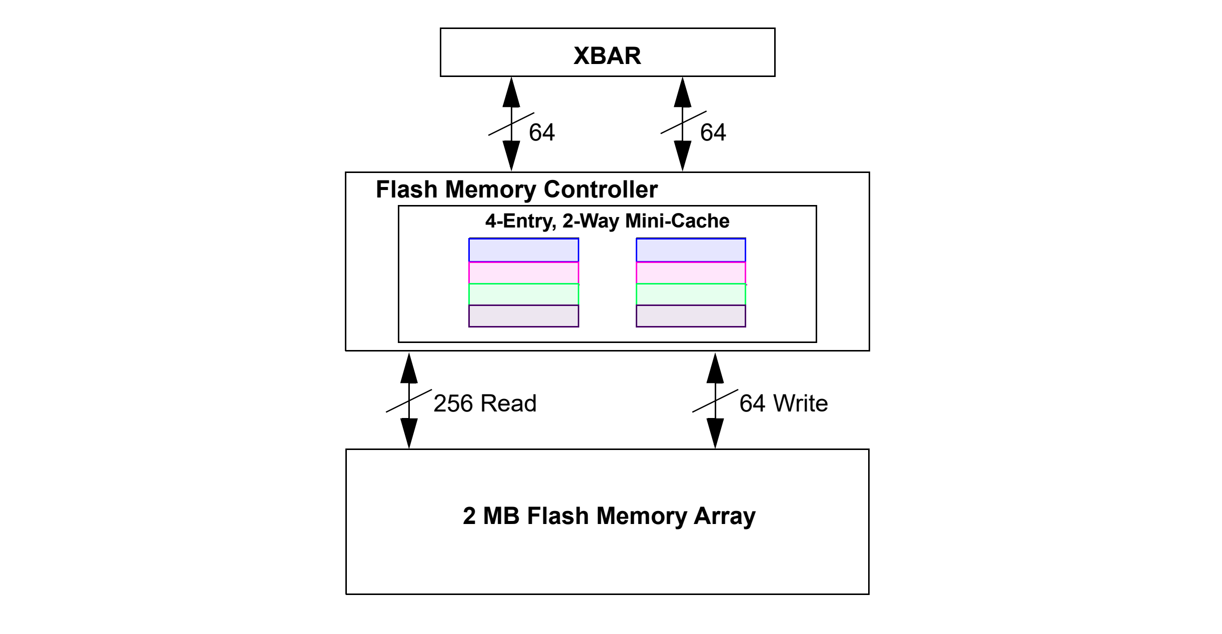
Each area of core RAM is accessible by bus mastering peripheral devices. Refer to the Memory Map chapter for the SPC584Bx SRAM map.

3.3 Embedded Flash memory

Flash memory on SPC584Bx device consists of one Flash memory controller and a Flash memory array module. The Flash controller provides Flash configuration and control functions and manages the interface between the Flash memory array and the device crossbar switch. *Figure 4* shows the memory architecture.

a. Refer to the respective module chapter for details.

Figure 4. SPC584Bx Flash memory block diagram



The remaining sections give the functional details of the Flash controller and Flash array, followed by the memory maps.

3.3.1 Flash memory controller

The SPC584Bx Flash memory controller acts as an interface between the system bus and the Flash memory array.

The Flash memory controller contains a four-entry, two-way set-associative mini-cache that delivers Flash read data with a zero-wait state response on lines that reside in the cache. Each entry contains one Flash read page, a 256-bit (32-byte) memory value. Read requests that miss the cache generate the needed Flash memory array access.

The Flash memory controller contains configuration registers that manage Flash functionality such as read buffering in the mini-cache, access control, and read wait state management of the Flash memory.

Refer to the Flash memory controller chapter for details.

3.3.2 Flash memory array

SPC584Bx device includes one single 2 MB Flash memory array, referred to as “main space,” plus an independent 16 KB block of one-time-programmable (OTP) Flash memory included to support systems that require non-volatile memory for security features or system initialization information. The independent 16 KB block is referred to as “UTEST space.”

For code Flash sectors, reads of the embedded Flash memory return a 256-bit page of data that may be buffered in the mini-cache in the Flash memory controller. Programming of the Flash may be done by quad word (128-bits), page (256-bits), or quad-page (1024-bits). Flash memory is erased on a block by block basis.

3.3.2.1 Features

- Flash segmentation
  - 16 KB, 32 KB, 128 KB, and 256 KB blocks provided for code or data
  - Four 16 KB blocks provided for EEPROM emulation
  - One 16 KB block and one 128 KB blocks provided for secure code
  - Two 16 KB blocks provided for secure data
  - Support for reading-while-writing when the accesses are to different partitions
- Flash memory protection: write protection and OTP function available for each block
- Test and initialization data stored in a non-volatile 16 KB OTP UTEST block
- ECC with triple bit detection, double and single bit correction
- Erase suspend, program suspend, and erase-suspended program all supported

3.3.2.2 Flash organization

The embedded Flash memory consists of six address spaces in four groupings:

- 16 KB, 32 KB and 128 KB low address spaces
- 16 KB mid address space
- 16 KB high address space
- 256 KB address space

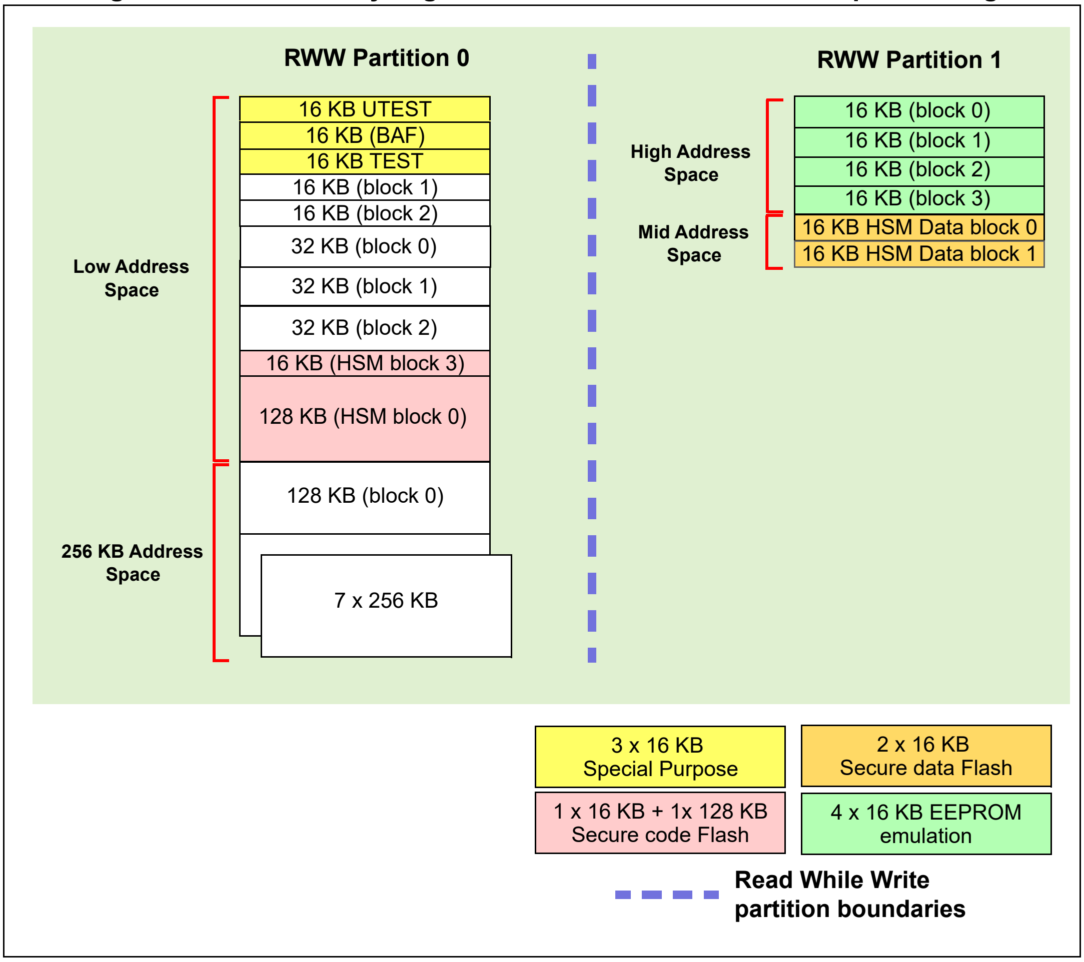
Address space group (low-, mid- or high-address space) determines which registers/fields are used to select blocks for modify operations or lock them from modify operations. Each address space is independent.

The Flash module is further divided into two partitions that determine locations for valid Read While Write (RWW) operations. While the embedded Flash memory is performing a “write” (program or erase) to a given partition, it can simultaneously read from the other partition.

- For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written).
- For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

Figure 5 shows the Flash block segmentation and Read While Write partitioning for SPC584Bx devices.

Figure 5. Flash memory segmentation and Read While Write partitioning



Note: Refer to the Memory Map chapter for the SPC584Bx Flash memory map.

3.3.2.3 UTEST memory space

Devices in the SPC584Bx family contain a 16 KB area of one-time-programmable (OTP) Flash memory for storing test information and device configuration data. *Refer to the Memory Map chapter* for the SPC584Bx UTEST memory map.

3.3.2.4 Chip-specific C40FMC Flash register information

Refer to *Chapter 35: Embedded Flash Memory* for chip-specific layouts of the on-chip Flash memory control registers that contain bits used to lock individual Flash blocks from accidental erase or select individual Flash blocks for operations.

3.4 End-to-end Error Correction Code (e2eECC)

To support market requirements related to improved functional and transient fault detection capabilities, this family of automotive microcontrollers includes end-to-end ECC (e2eECC*)* support. This e2eECC is structurally different from traditional “ECC at memory” functionality because it provides robust error detection capabilities from one endpoint of an information transfer to another endpoint, with temporary information storage in one or more intermediate components.

Memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect or correct errors that have occurred on data stored in the memory, or both. On the other hand, e2eECC generates error protection codes at the source of data generation. It sends the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master and performs a data integrity check using the previously stored error protection codes at a data memory when a read of stored information is requested by a bus master. The intermediate storage may transform the generated error protection codes into another format for storage and then regenerate the codes for provision when a request is made to read the stored information or it may simply store the original protection codes unaltered, depending on the particular unit.

Additionally, the error protection codes are generated on the basis of more than just the data associated with a storage location to protect additional information associated with an access. In particular, address information corresponding to the access location of the stored information is combined with the stored data at the data source to generate error protection codes that cover certain types of addressing errors in the system interconnect or in the memory unit. The error protection codes may be checked at the memory unit on a store to ensure that no address information or data has been corrupted while the request has transitioned through the device from the bus master source. The error protection codes may also be checked to ensure that address decoding within the storage memory was performed properly (although not all address decoding errors can be detected this way) or it may simply store the received data and protection codes at the address it receives.

On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location or locations and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory or in the retrieved data. This ensures that the address sent for the request was not corrupted, the addressed location was actually accessed (to the extent it is possible to ensure) and the stored data was error free, to the extent the ECC coding scheme is able to detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust than the previous implementation of locally generated and checked/corrected error protection at each memory unit.

A comparison of the traditional and e2eECC approaches is shown in *Table 3*.

Table 3. ECC comparison of data write-then-read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data and checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location

Table 3. ECC comparison of data write-then-read sequence (continued)

Traditional ECC	End-to-End ECC (e2eECC)
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

The scope of differences in the operations “covered” by the ECC checks is readily apparent with the e2eECC concept providing improved fault detection capabilities in two important aspects:

- The entire data transaction; from the initiating bus master, through the entire platform crossbar steering mechanism, to the destination slave target is covered during write accesses. Likewise, a read is checked from the initiating bus master, through the crossbar steering mechanism, through the actual memory read and transmission of the data plus checkbits back to the bus master, where the integrity and correctness of the entire transaction is checked.
- The selected ECC provides protection of both the address field as well as the data field, again for improved fault coverage.

Additionally, this particular structure also provides a significant implementation improvement. The traditional ECC at the memory approach places the checkbit decode and error syndrome generation with the error/no_error state, affecting whether the system bus transfer must be stalled (to correct a single or double bit error or report a noncorrectable event) or is allowed to complete (for error-free transfers) in a critical timing arc which often sets the upper limit of the operating frequency of the microcontroller. With the e2eECC scheme, the checkbit decode and error syndrome logic is located in the requesting bus master where there are generally more degrees of implementation freedom and the error/no_error state determination is removed from the system bus cycle termination logic; producing an improved timing arc and generally higher operating speeds.

Errors that occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in corrupted read data or checkbits being returned to a requester. While not all possible errors can be detected this way, this approach provides a substantial improvement versus traditional methods. Additional on-line diagnostics, or additional hardware, or both should be able to catch a majority of the errors not covered directly by the e2eECC scheme.

3.5 Security features

The SPC584Bx architecture implements a security framework that makes it possible to protect the micro-controller resources from potential attacks.

Note: Detailed information on the SPC584Bx security architecture is published in a separate document and is available on a limited basis to customers who have a non-disclosure agreement (NDA) with STMicroelectronics.

4 Signal Description

4.1 Production packages

The SPC584Bx device is available in these production packages:

- eTQFP64
- eTQFP100
- eTQFP144
- eLQFP176

Case numbers and outline drawings for each package are provided in the *SPC584Bx Microcontroller Data Sheet*.

4.2 Package pinouts/ballouts and pin/ball descriptions

Refer to the *SPC584Bx Microcontroller Data Sheet* and the *SPC584Bx pinout Microsoft Excel® workbook file attached to IO_Definition document*.

The *SPC584Bx pinout* provides signal descriptions and related information for the SPC584Bx device:

- the *I/O Signal Table sheet* contains information on generic pins, including the port pin, SIUL MSCR number, MSCR SSS, function, module, signal description, direction, pad type, and package pin numbers for each I/O pin
- the *Input Muxing sheet* contains the signal multiplexing options
- the *Supply Pins sheet* provides information on power supply and reference voltage pins
- the *Miscellaneous sheet* provides information on system pins (analog, reset, test, debug and trace pins)

The SIUL module is used to multiplex the device input/output pins and the internal input/output ports.

Note: Multi-function pins are programmable via their respective Multiplexed Signal Configuration Register (MSCR) Source Signal Select (SSS) values. The SSS value selects which source signal is connected to the associated destination.

5 Memory Map

Table 4 details the device memory map for the SPC584Bx.

All addresses on the SPC584Bx, including those that are reserved, are identified. The addresses represent the physical addresses assigned to each region or module name. A memory access, either read or write, to any region marked as “Reserved” will result in a Transfer Error. This error can generate an interrupt request in the CPU.

Table 4. Overview Memory Map

Start Address	End Address	Description
0x00000000	0x003FFFFF	Reserved
0x00400000	0x0FFFFFFF	Flash memory controller (refer to Table 5)
0x10000000	0x3FFFFFFF	Reserved
0x40000000	0x407FFFFF	System RAM (refer to Table 6)
0x40800000	0x4FFFFFFF	Reserved
0x50000000	0x5FFFFFFF	Processor local RAM (refer to Table 7)
0x60000000	0x9FFFFFFF	Reserved
0xA0000000	0xAFFFFFFF	Security module (refer to Table 8)
0xB0000000	0xF3FFFFFF	Reserved
0xF4000000	0xF7FFFFFF	Peripheral bridge 2 (refer to Table 9)
0xF8000000	0xFBFFFFFF	Peripheral bridge 1 (refer to Table 10)

5.1 Flash memory

Table 5 details the flash memory map for the SPC584Bx.

Table 5. Flash memory map

Start address	End address	Description	1 RWR partition RWW Partition ID
0x00000000	0x003FFFFF	Reserved	
Test & BAF Flash Blocks			
0x00400000	0x00403FFF	16 KB UTEST NVM Block Space(1)	0(2)
0x00404000	0x00407FFF	16 KB BAF block0	0(2)
0x00408000	0x0060BFFF	Reserved	
Security Code Flash Blocks			
0x0060C000	0x0060FFFF	16 KB HSM Code block3	0
0x00610000	0x0062FFFF	128 KB HSM Code block0	0

Table 5. Flash memory map (continued)

Start address	End address	Description	1 RWR partition RWW Partition ID
0x00630000	0x0067FFFF	Reserved	
Security Data Flash Blocks			
0x00680000	0x00683FFF	16 KB HSM EEPROM block0	1
0x00684000	0x00687FFF	16 KB HSM EEPROM block1	1
0x00688000	0x007FFFFF	Reserved	
Data Flash Blocks			
0x00800000	0x00803FFF	16 KB EEPROM Block0	1
0x00804000	0x00807FFF	16 KB EEPROM Block1	1
0x00808000	0x0080BFFF	16 KB EEPROM Block2	1
0x0080C000	0x0080FFFF	16 KB EEPROM Block3	1
0x00810000	0x00FBFFFF	Reserved	
Low & Mid Flash Blocks			
0x00FC0000	0x00FC3FFF	16 KB Code Flash block1	0
0x00FC4000	0x00FC7FFF	16 KB Code Flash block2	0
0x00FC8000	0x00FCFFFF	32 KB Code Flash block0	0
0x00FD0000	0x00FD7FFF	32 KB Code Flash block1	0
0x00FD8000	0x00FDFFFF	32 KB Code Flash block2	0
0x00FE0000	0x00FFFFFF	128 KB Code Flash block0	0
Large Flash Blocks			
0x01000000	0x103FFFF	256 KB Code Flash block0	0
0x01040000	0x0107FFFF	256 KB Code Flash block1	0
0x01080000	0x010BFFFF	256 KB Code Flash block2	0
0x010C0000	0x010FFFFF	256 KB Code Flash block3	0
0x01100000	0x0113FFFF	256 KB Code Flash block4	0
0x01140000	0x0117FFFF	256 KB Code Flash block5	0
0x01180000	0x011BFFFF	256 KB Code Flash block6	0
0x011C0000	0x0FFFFFFF	Reserved	

1. Refer to for additional details about how the UTest Flash Block is organized
2. Same Flash physical block

5.2 System memory

The following table details the system and stand-by RAM memory map.

Table 6. System and stand-by RAM memory map

Start offset	End offset	Assigned XBAR slave port	Description	Size
0x40000000	0x400A7FFF	Reserved		
0x400A8000	0x400A9FFF	PRAMC_2	System RAM 2 (Standby RAM)	8 KB
0x400AA000	0x400AFFFF		System RAM 2 (Standby RAM)	24 KB
0x400B0000	0x400C7FFF		System RAM 2 (Standby RAM)	96 KB
0x400C8000	0x407FFFFF	Reserved		
0xA0000000	0xAFFFFFFF	—	HSM (refer to *Table 8*)	

5.3 Local memory

Table 7 details the memory map for local RAM processor.

Table 7. Processor local RAM memory map

Start offset	End offset	Description	Used size
0x50000000	0x507FFFFF	Reserved	
0x50800000	0x5080FFFF	Reserved	
0x50810000	0x527FFFFF	Reserved	
0x52800000	0x5280FFFF	D-MEM CPU_2 (1)	64 KB
0x52810000	0x5FFFFFFF	Reserved	

1. Refer to *Table 1082: BAF memory organization* for the “BAF data area and stack” which uses a range of D-MEM CPU_2.

5.4 Security module memory

Table 8 details the memory map for the Hardware Security Module (HSM).

Table 8. HSM memory map

Start offset	End offset	Allocated size	Used size
0xA0000000	0xA0007FFF	32 KB	32 KB
0xA0008000	0xA0009FFF	8 KB(1)	8 KB (Stand-by domain)
0xA000A000	0xAFFFFFFF	Reserved	

1. Enabled during stand-by depending on a DCF bit.

5.5 Peripheral memory

Table 9 and *Table 10* list the memory map of the peripheral region. This region is sub-divided into 16 KB byte regions or 'slots'. A peripheral occupies 1 or more slots.

Within a 16 KB byte peripheral slot, it is normal for the peripheral registers to only occupy part of that 16 KB byte slot - normally from the lowest address upwards. The individual chapters for each block, define all the registers within that contiguous group of registers.

Table 9. Peripheral memory map (PBRIDGE 2)

Start Address	End Address	Peripheral	ON Platform Peripheral Slot[1]	OFF Platform Peripheral Slot[2]	Size[3] [Byte]
0xF4000000	0xF4003FFF	PBRIDGE_2	0		16384
0xF4004000	0xF4007FFF	Reserved			
0xF4008000	0xF400BFFF	XBAR_1	2		16384
0xF400C000	0xF400FFFF	Reserved			
0xF4010000	0xF4013FFF	XBIC_Concentrator_1	4		16384
0xF4014000	0xF4017FFF	SMPU_1	5		16384
0xF4018000	0xF401BFFF	Reserved			
0xF401C000	0xF401FFFF	XBIC_1 (for XBAR_1)	7		16384
0xF4020000	0xF4027FFF	Reserved			
0xF4028000	0xF402BFFF	PCM_0	10		16384
0xF402C000	0xF4033FFF	Reserved			
0xF4034000	0xF4037FFF	PFLASH_1	13		16384
0xF4038000	0xF4043FFF	Reserved			
0xF4044000	0xF4047FFF	INTC_1	17		16384
0xF4048000	0xF4057FFF	Reserved			
0xF4058000	0xF405BFFF	SWT_2 (core 2)	22		16384
0xF405C000	0xF405FFFF	SWT_3 (security)	23		16384
0xF4060000	0xF406FFFF	Reserved			
0xF4070000	0xF4073FFF	STM_2 (core 2)	28		16384
0xF4074000	0xF40A3FFF	Reserved			
0xF40A4000	0xF40A7FFF	eDMA_1	41		16384
0xF40A8000	0xF40ABFFF	PRAM_2	42		16384
0xF40AC000	0xF40E3FFF	Reserved			
0xF40E4000	0xF40E7FFF	TDM_0	57		16384
0xF40E8000	0xF7C2FFFF	Reserved			
0xF7C30000	0xF7C33FFF	BCTU_0		243	16384
0xF7C34000	0xF7C37FFF	STDBY_CTU_0		242	16384
0xF7C38000	0xF7C3BFFF	eMIOS_0		241	16384
0xF7C3C000	0xF7C53FFF	Reserved			

Table 9. Peripheral memory map (PBRIDGE 2) (continued)

Start Address	End Address	Peripheral	ON Platform Peripheral Slot(1)	OFF Platform Peripheral Slot(2)	Size(3) [Byte]
0xF7C54000	0xF7C57FFF	ETHERNET_0		234	16384
0xF7C58000	0xF7DFFFFF	Reserved			
0xF7E00000	0xF7E03FFF	SAR_ADC_12bit_0		127	16384
0xF7E04000	0xF7E33FFF	Reserved			
0xF7E34000	0xF7E37FFF	SAR_ADC_10bit_STDBY		114	16384
0xF7E38000	0xF7E3BFFF	Reserved			
0xF7E3C000	0xF7E3FFFF	SAR_ADC_12bit_B0 (Supervisor)		112	16384
0xF7E04000	0xF7E67FFF	Reserved			
0xF7E68000	0xF7E6BFFF	IIC_0		101	16384
0xF7E6C000	0xF7E6FFFF	Reserved			
0xF7E70000	0xF7E73FFF	DSPI_0		99	16384
0xF7E74000	0xF7E77FFF	DSPI_2		98	16384
0xF7E78000	0xF7E7BFFF	DSPI_4		97	16384
0xF7E7C000	0xF7E7FFFF	DSPI_6		96	16384
0xF7E80000	0xF7E8BFFF	Reserved			
0xF7E8C000	0xF7E8FFFF	LINFlexD_0		92	16384
0xF7E90000	0xF7E93FFF	LINFlexD_2		91	16384
0xF7E94000	0xF7E97FFF	LINFlexD_4		90	16384
0xF7E98000	0xF7E9BFFF	LINFlexD_6		89	16384
0xF7E9C000	0xF7E9FFFF	LINFlexD_8		88	16384
0xF7EA0000	0xF7EA3FFF	LINFlexD_10		87	16384
0xF7EA4000	0xF7EA7FFF	LINFlexD_12		86	16384
0xF7EA8000	0xF7ED3FFF	Reserved			
0xF7ED4000	0xF7ED7FFF	CAN_SUB_0_MESSAGE_RAM		74	16384
0xF7ED8000	0xF7EDBFFF	Reserved			
0xF7EDC000	0xF7EDFFFF	CAN_SUB_0_M_CAN_0		72	16384
0xF7EE0000	0xF7EE7FFF	Reserved			
0xF7EE8000	0xF7EEBFFF	CAN_SUB_0_M_CAN_1		69	16384
0xF7EEC000	0xF7EEFFFF	CAN_SUB_0_M_CAN_2		68	16384
0xF7EF0000	0xF7EF3FFF	CAN_SUB_0_M_CAN_3		67	16384
0xF7EF4000	0xF7F03FFF	Reserved			
0xF7F04000	0xF7F07FFF	CCCU_0		62	16384

Table 9. Peripheral memory map (PBRIDGE 2) (continued)

Start Address	End Address	Peripheral	ON Platform Peripheral Slot(1)	OFF Platform Peripheral Slot(2)	Size(3) [Byte]
0xF7F08000	0xF7F2FFFF	Reserved			
0xF7F30000	0xF7F33FFF	HSM Host I/F		51	16384
0xF7F34000	0xF7F37FFF	Reserved			
0xF7F38000	0xF7F3BFFF	DTS		49	16384
0xF7F3C000	0xF7F3FFFF	JDC		48	16384
0xF7F40000	0xF7F43FFF	Reserved			
0xF7F44000	0xF7F47FFF	STCU2		46	16384
0xF7F48000	0xF7F4BFFF	JTAGM		45	16384
0xF7F4C000	0xF7F4FFFF	Reserved			
0xF7F50000	0xF7F53FFF	MEMU		43	16384
0xF7F54000	0xF7F57FFF	IMA		42	16384
0xF7F58000	0xF7F63FFF	Reserved			
0xF7F64000	0xF7F67FFF	CRC_0		38	16384
0xF7F68000	0xF7F6BFFF	Reserved			
0xF7F6C000	0xF7F6C1FF	DMAMUX_0		36	512
0xF7F6C200	0xF7F83FFF	Reserved			
0xF7F84000	0xF7F87FFF	PIT_0		30	16384
0xF7F88000	0xF7F93FFF	Reserved			
0xF7F94000	0xF7F97FFF	RTC/API		26	16384
0xF7F98000	0xF7F9BFFF	WKPU		25	16384
0xF7F9C000	0xF7F9FFFF	Reserved			
0xF7FA0000	0xF7FA03FF	MC_PCU		23	1024
0xF7FA0400	0xF7FA3FFF	PMC_DIG			15360
0xF7FA4000	0xF7FA7FFF	Reserved			
0xF7FA8000	0xF7FABFFF	MC_RGM		21	16384
0xF7FAC000	0xF7FAFFFF	Reserved			

Table 9. Peripheral memory map (PBRIDGE 2) (continued)

Start Address	End Address	Peripheral	ON Platform Peripheral Slot(1)	OFF Platform Peripheral Slot(2)	Size(3) [Byte]
0xF7FB0000	0xF7FB003F	RC16M_DIG		19	64
0xF7FB0040	0xF7FB007F	RC1024K_DIG			64
0xF7FB0080	0xF7FB00BF	OSC40M_DIG			64
0xF7FB00C0	0xF7FB00FF	OSC32K_DIG (4)			64
0xF7FB0100	0xF7FB013F	PLL_DIG			64
0xF7FB0140	0xF7FB01FF	Reserved			
0xF7FB0200	0xF7FB023F	CMU_0 (PLL0, XOSC, IRCOSC)			64
0xF7FB0240	0xF7FB05FF	Reserved			
0xF7FB0600	0xF7FB3FFF	MC_CGM			14848
0xF7FB4000	0xF7FB7FFF	Reserved			
0xF7FB8000	0xF7FBBFFF	MC_ME		17	16384
0xF7FBC000	0xF7FBFFFF	Reserved			
0xF7FC0000	0xF7FC3FFF	SIUL2		15	16384
0xF7FC4000	0xF7FDFFFF	Reserved			
0xF7FE0000	0xF7FE3FFF	FLASH_0		7	16384
0xF7FE4000	0xF7FE7FFF	FLASH_ALT_0		6	16384
0xF7FE8000	0xF7FF3FFF	Reserved			
0xF7FF4000	0xF7FF7FFF	PASS		2	16384
0xF7FF8000	0xF7FFBFFF	SSCM		1	16384
0xF7FFC000	0xF7FFFFFF	BAR		0	16384

1. Refer to PACR registers in the Peripheral Bridge chapter
2. Refer to OPACR registers in the Peripheral Bridge chapter
3. This number indicates the assigned address range for the peripheral module on the peripheral bridge. The implemented memory size or the number of implemented peripheral registers might be smaller. Refer to the dedicated peripheral chapter or device configuration chapter for more details.
4. It will eventually be removed based on the results from Superset device validation

Table 10. Peripheral memory map (PBRIDGE 1)

Start Address	End Address	Peripheral	ON Platform Peripheral Slot(1)	OFF Platform Peripheral Slot(2)	Size(3) [Byte]
0xF8000000	0xF8003FFF	PBRIDGE_1	0		16384
0xF8004000	0xFBC37FFF	Reserved			

Table 10. Peripheral memory map (PBRIDGE 1) (continued)

Start Address	End Address	Peripheral	ON Platform Peripheral Slot(1)	OFF Platform Peripheral Slot(2)	Size(3) [Byte]
0xFBC38000	0xFBC3BFFF	eMIOS_1		241	16384
0xFBC3C000	0xFBDFFFFF	Reserved			
0xFBE00000	0xFBE03FFF	SAR_ADC_12bit_1		127	16384
0xFBE04000	0xFBE6FFFF	Reserved			
0xFBE70000	0xFBE73FFF	DSPI_1		99	16384
0xFBE74000	0xFBE77FFF	DSPI_3		98	16384
0xFBE78000	0xFBE7BFFF	DSPI_5		97	16384
0xFBE7C000	0xFBE8BFFF	Reserved			
0xFBE8C000	0xFBE8FFFF	LINFlexD_1		92	16384
0xFBE90000	0xFBE93FFF	LINFlexD_3		91	16384
0xFBE94000	0xFBE97FFF	LINFlexD_5		90	16384
0xFBE98000	0xFBE9BFFF	LINFlexD_7		89	16384
0xFBE9C000	0xFBE9FFFF	LINFlexD_9		88	16384
0xFBEA0000	0xFBEA3FFF	LINFlexD_11		87	16384
0xFBEA4000	0xFBEA7FFF	Reserved			
0xFBEA8000	0xFBEABFFF	LINFlexD_15		85	16384
0xFBEAC000	0xFBED3FFF	Reserved			
0xFBED4000	0xFBED7FFF	CAN_SUB_1_MESSAGE_RAM		74	16384
0xFBED8000	0xFBEE3FFF	Reserved			
0xFBEE4000	0xFBEE7FFF	CAN_SUB_1_M_CAN_1		70	16384
0xFBEE8000	0xFBEEBFFF	CAN_SUB_1_M_CAN_2		69	16384
0xFBEEC000	0xFBEEFFFF	CAN_SUB_1_M_CAN_3		68	16384
0xFBEF0000	0xFBEF3FFF	CAN_SUB_1_M_CAN_4		67	16384
0xFBEF4000	0xFBF57FFF	Reserved			
0xFBF58000	0xFBF5BFFF	FCCU		41	16384
0xFBF5C000	0xFBF63FFF	Reserved			
0xFBF64000	0xFBF67FFF	CRC_1		38	16384
0xFBF68000	0xFBF6BFFF	Reserved			
0xFBF6C000	0xFBF6C1FF	DMAMUX_1		36	512
0xFBF6C200	0xFBF83FFF	Reserved			
0xFBF84000	0xFBF87FFF	PIT_1		30	16384
0xFBF88000	0xFBFB01FF	Reserved			

Table 10. Peripheral memory map (PBRIDGE 1) (continued)

Start Address	End Address	Peripheral	ON Platform Peripheral Slot(1)	OFF Platform Peripheral Slot(2)	Size(3) [Byte]
0xFBFB0200	0xFBFB023F	CMU_1 (CORE, XBAR)		19	64
0xFBFB0240	0xFBFB027F	CMU_2 (HPBM)			64
0xFBFB0280	0xFBFB02BF	CMU_3 (PBRIDGE)			64
0xFBFB02C0	0xFBFB033F	Reserved			
0xFBFB0340	0xFBFB037F	CMU_6 (SARADC)			64
0xFBFB0380	0xFBFB047F	Reserved			
0xFBFB0480	0xFBFB04BF	CMU_11 (FBRIDGE)			64
0xFBFB04C0	0xFBFB04FF	CMU_12 (EMIOS)			64
0xFBFB0500	0xFBFB053F	Reserved			
0xFBFB0540	0xFBFB057F	CMU_14 (PFBRIDGE)			64
0xFBFB0580	0xFBFCFFFF	Reserved			
0xFBFD0000	0xFBFD3FFF	SIPI_1 (debug)		11	16384
0xFBFD4000	0xFBFD7FFF	Reserved			
0xFBFD8000	0xFBFDBFFF	LFAST_1 (debug)		9	16384
0xFBFDC000	0xFBFFFFFF	Reserved			

1. Refer to PACR registers in the Peripheral Bridge chapter
2. Refer to OPACR registers in the Peripheral Bridge chapter
3. This number indicates the assigned address range for the peripheral module on the peripheral bridge. The implemented memory size or the number of implemented peripheral registers might be smaller. Refer to the dedicated peripheral chapter or device configuration chapter for more details.

5.6 UTest memory

Table 11. UTest memory map

Start offset	End offset	Description
0x0000	0x0007	Temperature sensor calibration
0x0008	0x001F	Reserved
0x0020	0x002F	Factory Erase Diary Location
0x0030	0x003F	Test Mode Disable Override Passcode (1)
0x0040	0x004F	Test Mode Disable Seal (1)
0x0050	0x005F	Test Mode Disable Group A

Table 11. UTest memory map (continued)

Start offset	End offset	Description
0x0060	0x006F	Test Mode Disable Group B
0x00A0	0x00BF	UID
0x00C0	0x00CF	SW-DCF Start address(1)
0x00D0	0x00FF	Reserved
0x0100	0x011F	eFuse Bypass Password (2)
0x0120	0x013F	JTAG Password (3)
0x0140	0x015F	PASS password - Group 0 (3)
0x0160	0x017F	PASS password - Group 1 (3)
0x0180	0x019F	PASS password - Group 2 (3)
0x01A0	0x01BF	PASS password - Group 3 (3)
0x01C0	0x01DF	Reserved
0x01E0	0x01FF	Life Cycle slot 0 - ST Production
0x0200	0x021F	Life Cycle slot 1 - Customer Delivery
0x0220	0x023F	Life Cycle slot 2 - OEM Production
0x0240	0x025F	Life Cycle slot 3 - In Field
0x0260	0x027F	Life Cycle slot 4 - Failure Analysis
0x0280	0x029F	Customer Single Bit Correction area (4)
0x02A0	0x02BF	Customer Double Bit Correction area (5)
0x02C0	0x02DF	Customer Triple Bit Detection area (6)
0x02E0	0x02FF	Customer Programmable Detection Area
0x0300	0x0307	DCF Start Record(7)
0x0308	0x036F	Reserved
0x0370	0x0FFF	UTEST DCF Records(8)
0x1000	0x1FFF	Customer OTP data

1. The effective data is 4 bytes, but because of the new DEC ECC scheme, 16 bytes are still reserved.
2. Used to enable the Flash Test Mode when the eFuse protection is not active.
3. Access to this location is conditioned by the Life Cycle.
4. This area is written by ST during production phase with a specific content aimed to generate single bit errors, on behalf of diagnostic at application level of the ECC error correction path.
5. This area is written by ST during production phase with a specific content aimed to generate double bit errors, on behalf of diagnostic at application level of the ECC error detection path.
6. This area is written by ST during production phase with a specific content aimed to generate triple bit errors, on behalf of diagnostic at application level of the ECC error correction path.
7. The SSCM “SHD_DCF_OFFSET” parameter shall be aligned to the start of this area. Note that the UTest Flash block is seen by the SSCM at the following address: 0x0080_0000. The SSCM START RECORD DCF (refer to DCF start record in *Chapter 9: Device Configuration Format (DCF) Records*) must always be programmed at the beginning of this area. Because of the new DEC ECC scheme, the 64b following the START DCF record shall still be programmed in order to let the SSCM complete the DCF parsing (otherwise it would stop when a virgin record is found).

8. Factory can program some UTest DCF records during production test. Consider that the first free location may change. Refer to Sales to retrieve the information for the first free location or directly read the UTest to find it.

6 Functional safety

6.1 Introduction

The SPC584Bx offers a set of features to support applications that need to fulfill functional safety requirements as defined by automotive safety integrity level ASIL B of ISO 26262. This chapter gives an overview of the safety implementation for the SPC584Bx, and discusses the operation of the safety mechanisms.

6.2 Safety overview

The SPC584Bx has been designed for automotive applications executing safety-relevant tasks that require a fail-silent or fail-indicate MCU. The SPC584Bx supports a Fault Tolerant Time Interval (FTTI) of at least 10 ms. Shorter intervals are possible depending on the actual combination of failure detection and signaling mechanisms but the SPC584Bx generally supports an FTTI of 10 ms.

6.3 Module categorization

The definition of the SPC584Bx safety concept and its validation (particularly the computation of Probabilistic Metric for Random Hardware Failures (PMHF)) has been done by splitting system modules into four categories as shown in *Table 12*.

Table 12. PMHF module categories

Module category	Category abbreviation	Description
Vital system modules	ViMo	All modules that can directly influence the correct operation of the software execution (by influence of the correct function of CPU, RAM, Flash, and the bus interconnect). Examples of ViMo are the core, the cache, the crossbar, and the interrupt controller (INTC), as well as clock or power generation.
Peripheral modules	PeMo	All microcontroller modules involved in I/O operation, specifically including the I/O bridge.
Non-safety modules	NoSaMo	Modules not used to implement any safety related functionality (for example, debug modules), modules which are functionally similar to modules in the categories above but do not provide any embedded safety provisions by default or are not used for safety in a use case scenario. In case they are used in a safety-relevant way, application measures are necessary to ensure their safe operation. Notice that interference-freedom is guaranteed in both cases.
Other support modules	SuMo	These are modules that, although safety-relevant and in principle potential members of the ViMo category, cannot lead to a violation of the safety goal on their own. Typically, these will only have multiple point faults, but not single point faults. Example SuMo’s are the RCCU and Clock Monitoring Unit (CMU).

Table 13 shows module classifications for the SPC584Bx.

Table 13. Module classification

Classification	Modules
ViMos	Core_2 with controllers for I-Cache, D-Cache and local memories, I-Cache, D-Cache and D-Mem arrays, eDMA, DMACHMUX, Concentrator DMA, INTC, XBAR, SRAM controller, System RAM array, Flash memory controller, Flash memory array,System timer, MC_RGM, Dual PLL Digital, XOSC, MC_CGM, MC_ME, Voltage regulator, MC_PMC, STCU2, SSCM, PIT
PeMos	SAR ADC, LINFlexD, DSPI, M_CAN, Ethernet, SIUL2, XBAR_1 (Peripheral), Concentrator Ethernet, LFAST, PBRIDGE*n*
NoSaMos	External memory interface (LFAST, SIPI), HSM, Nexus, JTAG, Concentrator HSM, I2C, WKPU, SPU, STANDBY SAR ADC

Table 13. Module classification (continued)

Classification	Modules
SuMos	CRC units, Temperature Sensors, IRCOSC, CMUs, HVDs, Bandgap reference, IMA, REG_PROT, SWT, FCCU, MEMU

The SPC584Bx implements no specific protection of NoSaMos against Common Cause Failures between them.

6.4 System implementation

6.4.1 General concept

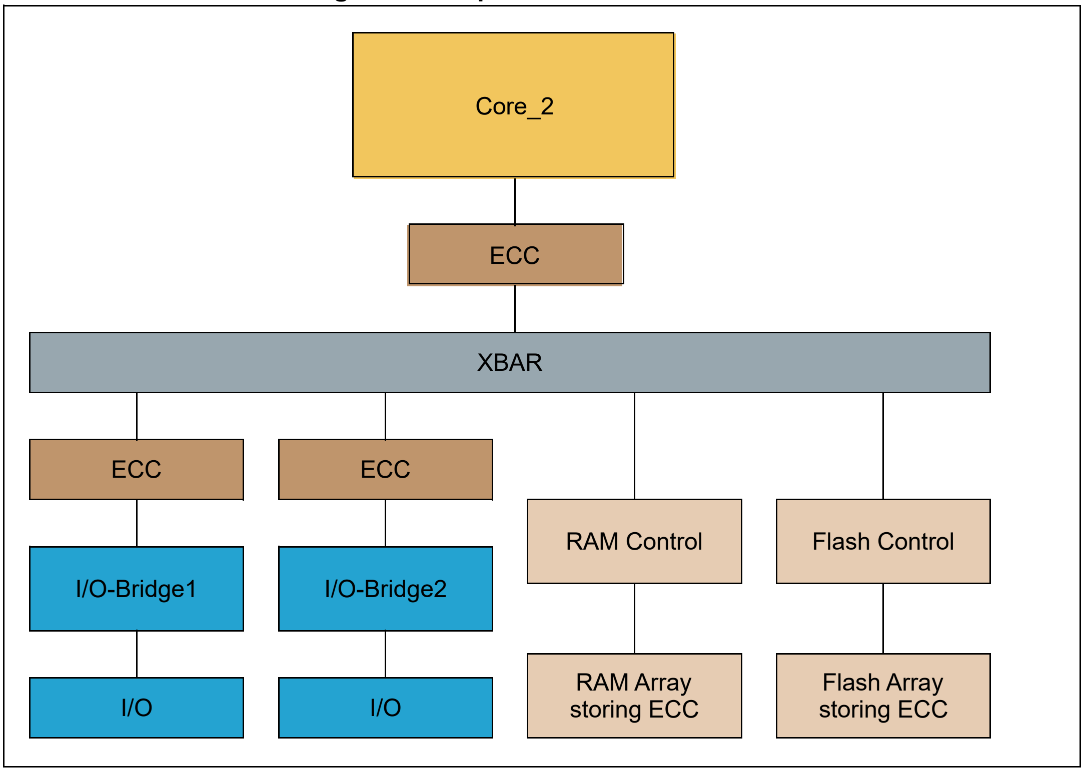
In general, safety integrity is achieved in the following ways:

- The safety of storage and of the data path to storage and periphery is ensured by End-to-End ECC (E2E ECC) with address encoding and selected additional measures for individual modules. For the periphery, end-to-end ends at the I/O bridges (refer to *Section 6.4.1.1: End-to-End protection on data path*).
- Clock and power, generation and distribution, are supervised by dedicated monitors (refer to *Section 6.4.1.2: Clock and power monitoring*).
- The safety of the periphery is ensured by application-level measures (such as connecting one sensor to different I/O modules, sensor validation by sensor fusion, and so on). Hardware supports this application-level redundancy by providing redundant I/O modules connected to different peripheral bridges (PBRIDGE) to maximize the independence between the monitored and monitoring resources (refer to *Section 6.4.1.3: I/O peripherals*, and *Section 6.4.1.4: Communication controllers*).
- The Fault Collection and Control Unit (refer to “Fault Collection and Control Unit (FCCU)” chapter for FCCU details) is responsible for collecting and reacting to failure notifications (refer to *Section 6.4.1.5: FCCU and failure monitoring*). MEMU (refer to *Section 6.4.3.4: Memory Error Management Unit (MEMU)*) is responsible for the collection and reporting (to the FCCU) of error events in system memories, and Flash memory, as well as E2E ECC errors caused by the XBAR, or RAM or Flash memory controllers. Error events include ECC corrections and detections as well as faults detected by other embedded monitors.
- Common Cause Failures (CCF) are dealt with by a set of measures for both control and avoidance of CCFs spanning system-level approaches (such as temperature and nonfunctional signal monitoring) and back-end techniques (such as isolated silicon areas and routing constraints) (refer to *Section 6.4.2: Common Cause Failure measures*).

6.4.1.1 End-to-End protection on data path

Connections between XBAR masters and slaves (clients) are denoted as data paths. Data corruption on all data paths between the Safety Core and any client is detected with at least 99% coverage via two main safety mechanisms: data from the replicated cores is encoded using Error Correcting Code (ECC), which is implemented with a single-error correction, double-error detection (SECDED) code with a Hamming distance of 4 and includes coverage of addressing information; control signals and address decoding are monitored to verify the data reaches all of the intended clients, from all possible connections to these clients and the intended operation is performed on the target address. *Figure 6* shows a general view of ECC schema.

Figure 6. Simplified view of E2E ECC



Note: Specific implementation for the SPC584Bx varies depending on the special requirements of RAM and Flash memory concerning ECC handling.

ECC bits are generated on writes by XBAR masters and checked on reads. The ECC correction bits are stored alongside the data in Flash memory and RAM so, in principle, no ECC logic is necessary at the memories themselves. For this reason the ECC schema is referred to as End-to-End ECC (E2E ECC) in the following sections. For XBAR slaves, apart from memories, new ECC logic is added as these clients cannot store or produce the ECC correction bits. This resolves the problem where ECC needs to be calculated in real time before entering or exiting the ECC-protected data path. This setup leaves the data path downstream of the ECC units (for example, starting at, and downstream of, the peripheral bridges) unprotected by ECC. To detect corruptions introduced on those unprotected data paths, the PBRIDGEs and peripherals can be used redundantly (refer to *Section 6.4.1.3: I/O peripherals*), or other mechanisms may be used to validate peripheral operation.

The E2E ECC schema provides high detection capabilities against failures affecting the data content of the transaction. The inclusion of the target address in the computation of the redundancy bits (8 ECC bits) does allow the partial detection of addressing faults as well. To reach the desired integrity level, additional dedicated safety mechanisms are implemented in the data path particularly to:

- Improve the detection capability over addressing failures (no/multiple/wrong address selected), considering faults affecting address transmission (from master to client) as well as the decoding of the address.
- Provide coverage for control failures affecting, for example, the type (read vs. write) or size of a transaction.

Though safety mechanisms protecting the XBAR, the RAM controller, or the Flash memory controller are different, they are all based on the feedback of address and control information from the target to the source of the transaction, which is responsible for checking for consistency with respect to the intended transaction. Depending on the portion of the data path covered by the specific safety mechanism, the source can be an XBAR master port rather than the XBAR interface of the RAM or Flash controllers; the target is respectively an XBAR slave port, the RAM array, or the Flash module (refer to “Crossbar Switch (XBAR)”, “Flash memory controller”, and “RAM controller (PRAM)” chapters).

6.4.1.2 Clock and power monitoring

6.4.1.2.1 Clock

Clocks in the SPC584Bx are supervised by Clock Monitor Units (CMUs). The CMUs are driven by the IRCOSC (16 MHz internal oscillator) for independent operation from the monitored clocks. CMUs flag errors associated with conditions due to loss of lock, clock out of a programmable bound (over and under frequency), and loss of reference. If a supervised clock leaves the specified range for the device, a signal is sent to the FCCU.

6.4.1.2.2 Power

There are four types of voltage supervisors on the SPC584Bx, Low Voltage Detect (LVD), High Voltage Detect (HVD), Upper Voltage Detector (UVD) and Minimum Voltage Detector (MVD) monitors. Safety-relevant voltages are supervised for voltages that are out of these ranges. Since safety relevant voltages have the potential to disable the failure indication mechanisms of the MCU (such as FCCU, pads, and so on) the indication of these errors can be used to cause a direct transition of the MCU into the safe state (reset assertion) (refer to “Power management” and “Power Control Unit (MC_PCU)” chapters for details on power monitoring).

6.4.1.3 I/O peripherals

To allow a safety application to make redundant use of all I/O peripherals, each peripheral has at least two instances, and each instance is connected to a different PBRIDGE. This means, for example, that if DSPI is provided by the MCU, two DSPI modules (DSPI*n*, DSPI*m*) are included and connected externally through different pins. Internally, DSPI*n* would then be connected to PBRIDGE_0 and DSPI*m* to PBRIDGE_1 and be accessible via different addresses.

The arrangement of I/O peripherals onto two PBRIDGEs, as well as further CCF prevention measures, allow redundant use of peripherals while limiting possible causes of CCFs. Redundant usage includes usage of equivalent peripherals in a replicated way as well as usage of functionally different peripherals in, for example, feedback measurement loops.

Comparison of redundant operation is under the responsibility of the application software, and not the safety hardware mechanism.

6.4.1.4 Communication controllers

Communication controllers provide the ability to exchange information with external components and therefore fall under the same safety reasoning as I/O peripherals. Yet we assume that for high bandwidth communication controllers additional software measures are employed that do not require redundant communication peripherals.

The following communication controllers do not contain special safety mechanisms (above what is included in them by their protocol specifications) nor are they duplicated or spread over the PBRIDGE (in contrast to other I/Os, refer to *Section 6.4.1.3: I/O peripherals*):

- Ethernet
- M_CAN

Typically, software measures for the communication controllers (also called fault-tolerant communication layer) could contain E2E CRC data protection, sender identification, sequence numbering, and an acknowledgement mechanism.

6.4.1.4.1 Disabling of communication controllers

In the event of a dangerous failure, the MCU offers the capability of disabling transmission of individual channels of communication controllers (such as CAN modules). This prevents the transmission of erroneous messages while preserving the capability of communicating over the diagnostic bus. Disabling outputs is controlled by resetting the SIUL2_MSCR*n*[SMC] for the pins that are associated with communication controllers where this feature is needed (refer to the “System Integration Unit Lite2 (SIUL2)” chapter for details on disabling communication signals).

FCCU can drive a fault state on FI[0] with either one of the following two mechanisms:

- Error pin reaction due to a fault
- Software writing FCCU_CFG[FCCU_SET_CLEAR] = 01b

When the FCCU drives FI[0] to a fault state, the SIUL2 disables the output drivers of all pins whose SIUL2_MSCR*n*[SMC] = 0. This disables the possible transmission of erroneous messages. As long as the FCCU drives FI[0] to faulty state, the communication controllers’ Tx pins are disabled and internal pull-up are enabled. As soon as FI[0] is driven to non-faulty state, the communication controllers’ Tx pins are automatically enabled and state of internal pull-up/down restored.

The application should configure SIUL2_MSCR*n*[SMC] for pins that have active mapping of CAN modules (M_CAN) functionality and ensure that pin does not stay in undriven state.

Note: The FCCU uses an internal signal to disable the communication controller transmission, so transmission is disabled even when the FCCU cannot drive the FI[0] pin because the pin is configured as GPIO. Also, if FI[0] is used as error input, externally pulling it down will disable the communication controller transmission if this event drives FCCU to a fault state.

Disabling of communication controllers occurs on the Tx pins of the communication controller (for example, M_CAN) of the MCU even while the communication controller continues operating. Failed transmissions, when detected by the communication controller, may lead to additional error events.

6.4.1.5 FCCU and failure monitoring

The FCCU offers a hardware mechanism to aggregate error notifications and a configurable means to bring the device to a safe state. No CPU intervention is required for collection and control operation. Error indications are passed from the individual hardware components to the FCCU where the appropriate action is decided (according to the FCCU configuration).

6.4.1.5.1 External error indication

Failure of the MCU is signaled to one or two pins, FI[0] (ERROR0) and FI[1] (ERROR1). FI[0] can also serve as an error input mechanism (refer to “Fault Collection and Control Unit (FCCU)” chapter, and “FCCU configuration” section of the “Device configuration” chapter, for details on the fault output signals). Both pins can be multiplexed with other functions with different defaults.

The error indication on pins FI[0] and FI[1] is controlled by SIUL2 and FCCU. The mapping of the FCCU output signals on the relevant pins and the associated electrical characteristics are enabled by the SIUL2_MSCR[*n*] register (refer to table “SIUL2_MSCR0 - SIUL2_MSCR511 field description” in the “System Integration Unit Lite2 (SIUL2)” chapter for signal mapping details). Once the FCCU is mapped onto the pin, the logic level is driven by the FCCU.

FI[0] is controlled by the FCCU at startup by default, but the state of FI[1] is like other I/O signals (refer to table “Pin startup and reset states” in chapter “Signal Description” for default states of FI[0] and FI[1]).

Since it is possible to reset FCCU and SIUL2 independently, the behavior of FI[0] and FI[1] is the result of the combined state of SIUL2_MSCR[*n*] and the FCCU. (refer to section “Reset interface” in chapter “Fault Collection and Control Unit (FCCU)” for FCCU reset details, and chapter “Reset and Boot” for the state of the FCCU relative to the system during different reset phases).

During functional reset, the application shall consider only the PAD PB[11] as FCCU error out. The FCCU EOUT1, that is PC[2], is not enabled during functional reset.

The FCCU_STAT[ESTAT] error status flag can be read to show whether the FCCU is in an error state. This flag can be written by software to either a 1 (fault) or 0 (operational) when the FCCU is in operational state.

FCCU_EINOUT(EINO): The value of this register depends on the state of the pad. If the IBE functionality of the SIUL is enabled, the register bit reflects the physical status of the pin as seen by the input buffer. In case the protocol selected for EOUT interface is a toggling one (Dual rail or Time switching), this may not be true due to synchronization in the safety clock domain.

The register reflects the physical status of the pin from application point of view (external status).

FCCU_STAT(PhysicalErrorPin): This register field indicates the actual status of the Error pin as driven by FCCU to the pad based on different FCCU configurations and state of the failure input channels. The value reported in the FCCU_STAT (PhysicalErrorPin) could be different from what is seen at the pad.

6.4.1.5.2 Failure handling

The FCCU is an autonomous module that is responsible for reacting to failure indicators. A different reaction can be configured for each failure source. Overall failure reaction time requires time for detecting, processing, and indicating the error. During this time, the SPC584Bx could provide wrong results to the system.

Failure sources include:

- All failure indication signals from modules within the MCU.
- Control logic and signals monitored by the FCCU itself (refer to chapter “Fault Collection and Control Unit (FCCU)”).
- Software-initiated failure indications. For example, software signals the FCCU that it has evidence of a failure. Keep in mind that software can also directly influence the state of the F*n* pins.
- External failure input.

Available failure reactions are:

- Assertion of an interrupt (maskable or non-maskable)
- Resetting the MCU
- Changing the state of the failure indication pins, ERROR*n*
- Disabling the transmission capabilities of communication controllers (CAN, Ethernet. Note that, it is only possible in conjunction with changing the state of the failure indication pins
- No reaction

Software can read the failure source that caused a fault, either before or after a functional reset (the condition indicators are not volatile). Software can also reset the failure, but the external failure indication will stay in failure mode for a configurable minimum time. If necessary, software can also reset the MCU.

6.4.1.5.3 FCCU failure inputs from other modules

Table 14 shows the different failure input signals of the FCCU.

Table 14. FCCU failure inputs

FCCU channel	Source	Failure	Failure description	HW failure behaviour
0	PMC DIG	TEMP_ERROR	Temperature detector	Pulse signal. No SW intervention needed to clear it.
1	PMC DIG	LVD_ERROR	Voltage out of range from LVDs (non-destructive reset)	Pulse signal. No SW intervention needed to clear it.
2	PMC DIG	HVD_ERROR	Voltage out of range from HVDs (non-destructive reset)	Pulse signal. No SW intervention needed to clear it.
3	PMC DIG	DPMC_DCF_SAFETY_ERR	Digital PMC DCF Safety Error	Persistent till error negation

Table 14. FCCU failure inputs (continued)

FCCU channel	Source	Failure	Failure description	HW failure behaviour
4	PMC DIG	DPMC_VD_BIST	Digital PMC voltage detector BIST	Clear status through FCCU
5	SSCM/ FLASH 0	SSCM_XFER_FLASH_ERR	SSCM transfer error (during the SSCM and STCU DCF loading) + Flash memory initialization error	SSCM gets cleared by SW. fatal error pulse no SW needed. Memory DCF part is persistent till error negation.
6	STCU	STCU_UF	BIST result - wrong signature (STCU Unrecoverable Fault)	Clear status through FCCU
7	STCU	STCU_RF	BIST result - wrong signature (STCU Recoverable Fault)	Clear status through FCCU
8	STCU	STCU_LMBIST_USR_ERR	STCU fault in case MBIST control signals go to wrong condition during user application.	Clear status through FCCU
9	GLUE LOGIC	SPURIOUS_DEBUG_SSCM_ACTIVATION	JTAG or NPC not in reset or activation of dangerous debug functionality Spurious activation of SSCM	Clear status through FCCU
15	PLATFORM	DMA1 TCD EDC after ECC	DMA1 TCD EDC after ECC	Clear status through FCCU
16	Flash	Flash Reset Error	The pin/flag signals the following unrecoverable errors: – ECC errors on Flash internal reads during configuration loading (startup) – ECC errors on Flash internal reads during fw copy (startup) – Double ECC errors on KRAM during internal self-check routine (always running)	Cleared by reset only

Table 14. FCCU failure inputs (continued)

FCCU channel	Source	Failure	Failure description	HW failure behaviour
17	PLATFORM	SWT3_ResetReq	Reset request from SWT_3. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT3 (after a functional reset)	Cleared by reset only
18	PLATFORM	SWT2_ResetReq	Reset request from SWT_2. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT2 (after a functional reset)	Cleared by reset only
21	MEMU	MEMU_RAM_CE	MEMU RAM correctable error	Clear status in MEMU
22	MEMU	MEMU_RAM_UCE	MEMU RAM uncorrectable error	Clear status in MEMU
23	MEMU	MEMU_RAM_OV	MEMU RAM overflow(1)	Clear status in MEMU
24	MEMU	MEMU_PER_CE	MEMU Peripheral correctable error	Clear status in MEMU
25	MEMU	MEMU_PER_UCE	MEMU Peripheral uncorrectable error	Clear status in MEMU
26	MEMU	MEMU_PER_OV	MEMU Peripheral overflow(2)	Clear status in MEMU
27	MEMU	MEMU_FLS_CE	MEMU Flash correctable error	Clear status in MEMU
28	MEMU	MEMU_FLS_UCE	MEMU Flash uncorrectable error	Clear status in MEMU
29	MEMU	MEMU_FLS_OV	MEMU Flash overflow	Clear status in MEMU
30	IMA	IMA SoC	IMA SoC Active The fault is cleared by disabling IMA (IMA.SLCT.R = 0x0), followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Persistent till error negation
32	PLATFORM / SMPU	SMPU_MONITOR XBAR 1	SMPU XBAR 1 Monitor incorrectly refuses an access	Pulse signal. No SW intervention needed to clear it.

Table 14. FCCU failure inputs (continued)

FCCU channel	Source	Failure	Failure description	HW failure behaviour
34	PLATFORM / SMPU	SMPU ERROR XBAR 1	SMPU XBAR 1 refuses an access correctly.	Pulse signal. No SW intervention needed to clear it.
48	PLATFORM / CORE	DMA 1 TCD Ram feedback checker	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
49	PLL DIG	FM_PLL_0	PLL0 Loss of Lock (Interrupt event)	SW clear required in PLLDIG
50	PLL DIG	FM_PLL_1	PLL1 Loss of Lock (Interrupt event)	SW clear required in PLLDIG
51	CMU	CMU_0_XOSC	XOSC less than IRC, XTAL Pin Floating, EXTAL Pin Floating(3)	Clear status in CMU
52	CMU	CMU_0_PLL0	Sysclk frequency out of range(4)	Clear status in CMU
53	CMU	CMU_Platform	Monitoring internal clocks of comp_subsys(5)	Clear status in CMU
54	CMU	CMU_other	Monitoring internal clocks of other clocks(6)	Clear status in CMU
56	PLATFORM / XBAR 1	XBAR 1 XBIC + XBAR 1 decoder checker	Indication of a hardware fault resulting in corrupted transaction through the XBAR or gaskets	Clear status through FCCU
57	PLATFORM / CORE	CORE 2 D-MEM address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
58	PLATFORM / CORE	CORE 2 D-MEM address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
59	PLATFORM / CORE	CORE 2 I-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU

Table 14. FCCU failure inputs (continued)

FCCU channel	Source	Failure	Failure description	HW failure behaviour
60	PLATFORM / CORE	CORE 2 I-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
61	PLATFORM / CORE	CORE 2 I-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
62	PLATFORM / CORE	CORE 2 I-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
63	FLASH 0	FLASH_REF_ERR	If the current and voltage Read references are out of range, this bit is set to signify that previous reads requested may have been corrupted.	The recommended clear mechanism is device reset
64	PLATFORM / FLASH 1	EDC_ECC_FLASH 1	EDC after ECC for FLASH array - Indication of a hardware fault in the ECC of the Flash memory resulting in corrupted ECC detection/correction.	Clear status through FCCU
65	PLATFORM / FLASH 1	EDC_ECC_FLASH_C 1	EDC after ECC for FLASH controller – Indication of hardware fault in the Flash memory controller resulting in corrupted ECC detection / correction.	Clear status through FCCU
66	PLATFORM / FLASH 1	ENC_ERR_FLASH 1	Flash encoding error – Indication of hardware fault resulting in corrupted Flash memory access.	Clear status through FCCU

Table 14. FCCU failure inputs (continued)

FCCU channel	Source	Failure	Failure description	HW failure behaviour
67	PLATFORM / FLASH 1	ADDR_FDBK_ERR_FLASH_C 1	PFlashC address feedback error - alarm indicating the Flash controller detected a transaction monitor mismatch when compared to the Flash safety feedback outputs	Clear status through FCCU
69	PLATFORM / CORE	CORE 2 I-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
70	PLATFORM / CORE	CORE 2 D-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
71	PLATFORM / CORE	CORE 2 D-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
72	PLATFORM / CORE	CORE 2 D-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
73	PLATFORM / CORE	CORE 2 D-Cache address ERR	Feedback checkers implemented in the platform covering memory address/CS/WE feedback signals	Clear status through FCCU
74	PLATFORM / PRAM 2	ADDR_FDBK_ERR_PRAM 2 + RAM_LWB_ERR	System RAM address feedback + RAM Late-Write buffer mismatch. Indication of an addressing error in system RAM or a write error in the RAM controller resulting in corrupted RAM access	Clear status through FCCU

Table 14. FCCU failure inputs (continued)

FCCU channel	Source	Failure	Failure description	HW failure behaviour
75	PLATFORM / PRAM 2	EDC_ERR_PRAM 2	EDC after ECC for RAM (read-modify-write ECC). Indication of a fault in the controller PRAM of the system RAM resulting in a potentially corrupted RAM word during RMW access or stale word during read.	Clear status through FCCU
78	TCU	DFT1	Test circuitry Group x activation	Persistent till reset
79	TCU(7)	DFT2	Test circuitry Group x activation	Persistent till reset
80	TCU	DFT3	Test circuitry Group x activation	Persistent till reset
81	TCU	DFT4	Test circuitry Group x activation	Persistent till reset
84	PLATFORM / CORE	Core 2 Exception	Core Machine check exception indication	Pulse signal. No SW intervention needed to clear it.
89	PLATFORM / PBRIDGE	PBRIDGE_1_e2eEDC_ERR	ECC bits in the PBRIDGE transaction contain an error	Persistent till error negation
91	PLATFORM / PBRIDGE	PBRIDGE_2_e2eEDC_ERR	ECC bits in the PBRIDGE transaction contain an error	Persistent till error negation
92	PLATFORM	Monitor_Concentrator1_ Alarm	Failure of Concentrator 1 Monitor	Clear status through FCCU
93	PLATFORM	Monitor_Concentrator_DMA	Concentrator DMA gasket monitor failure	Clear status through FCCU
94	MC_RGM	Safe Mode(8)	Safe mode entry request from RGM	Persistent till system remains in Safe mode
95	COMPENSATION CELLS	Compensation Disable	Indication of undesired deactivation of pad compensation. Fault may affect a distorted transmissions on I/Os associated. The indication is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm](9)	Persistent till error negation from HW

Table 14. FCCU failure inputs (continued)

FCCU channel	Source	Failure	Failure description	HW failure behaviour
96	GLUE LOGIC	EIN_ERR	Error input pin (from the external world)	Persistent till error negation from HW
101	PLATFORM	DSMC_2 monitor error	DSMC_2 monitor error	Persistent till error negation from HW
106	PLATFORM / CORE	CORE 2 I-MEM ILLEGAL ACCESS	CORE 2 Attempt to access non-existent I-MEM	Clear status through FCCU
107	PLATFORM	Monitor_Concentrator_HSM	Concentrator HSM gasket monitor failure	Clear status through FCCU
108	PLATFORM	Monitor_Concentrator1_ XBIC	Concentrator 1 XBIC monitor failure	Clear status through FCCU
109	PLATFORM	Monitor_Concentrator_DMA	Concentrator DMA ECC monitor failure	Clear status through FCCU
110	PLATFORM	Monitor_Concentrator_HSM	Concentrator HSM ECC monitor failure	Clear status through FCCU
111	PLATFORM	Monitor_Concentrator1_ SIPI_1	Concentrator 1 ECC monitor failure - SIPI	Clear status through FCCU
113	PLATFORM	Monitor_Concentrator1_ ETHERNET_0	Concentrator 1 ECC monitor failure - Ethernet	Clear status through FCCU
114	PLATFORM	Monitor_Concentrator_DMA	Concentrator DMA gasket monitor failure	Clear status through FCCU

1. Aggregate of RAM Correctable Error overflow flag, RAM Uncorrectable Error overflow flag, RAM Error buffer Overflow Flag
2. Aggregate of Peripheral RAM Correctable Error overflow flag, Peripheral RAM Uncorrectable Error overflow flag, Peripheral RAM Error Buffer Overflow Flag
3. CMU_0_PLL0_XOSC_IRCOSC OLR event + XTAL pin floating + EXTAL Pin Floating
4. CMU_0_PLL0_XOSC_IRCOSC FLL and FHH events
5. CMU_1_CORE_XBAR, CMU_2_HPBM, CMU_3_PBRIDGE FLL and FHH events, CMU_11_FBRIDGE FLL and FHH events, CMU_14_PFBRIDGE
6. CMU_6_SARADC, CMU_12_EMIOS
7. In the event of activation of RF[94] (that is safe mode request), RF[79] is also activated. If the RGM trigger of safe mode is functionally used, during servicing of the response to activation of the RF[79], software should check that the device is not in safe mode to discriminate between an unwanted activation of test circuitry and a safe mode request.
8. In the event of spurious activation of this fault source, RF[79] is activated together with RF[94]. In scenarios where safe mode trigger from RGM is a valid functional use case, the RF[94] should be disabled.
9. Compensation cell reduces the spread of some circuit parameters (slew rate of the output signal and the output impedance) in the IO buffers over temperature, process and voltage. Compensation cell might remain disabled until supplies have reached LVD290 threshold.

6.4.1.6 Operational interference protection

Being a multi-master system, SPC584Bx provides safety mechanisms to prevent non-safety masters from interfering with the operation of the Safety Core, as well as mechanisms to handle the concurrent execution of software with different (lower) ASIL. Interference freedom is guaranteed via a hierarchical memory protection schema including:

- MPUs
- PBRIDGEs
- Register protection

There are two Memory Protection Unit levels included in the SPC584Bx. The Core Memory Protection Unit (CMPU) is a mechanism included in each core to protect address ranges against access by software developed according to lower ASIL. The CMPU is typically used by the operating system to ensure inter-task interference protection.

The second MPU level is provided by the System MPUs (SMPU) located in each XBAR. The MPUs prevent access of different bus masters to address ranges and are typically used by the safety application to prevent NoSaMos (such as the Performance Core) to access the application’s safety-relevant resources.

Furthermore, the PBRIDGE can restrict read and write access to individual I/O modules based on the origin of the access and its state (user mode/supervisor mode).

Finally, the register protection included allows individual register to be locked against any manipulation without unlocking.

These safety mechanisms are further described in the chapters System Memory Protection Unit (SMPU), Peripheral Bridge (PBRIDGE) and Registers Protection (REG_PROT).

6.4.1.7 FCCU supervision (FOSU)

As the FCCU is a central component in reacting to errors, it is itself supervised even though an error in it can only cause a latent failure. This supervision is provided by the FCCU Output Supervision Unit (FOSU). The FOSU receives failure indications at the same time as the FCCU. Unless the respective failure is switched off, the FOSU will observe the outputs of the FCCU (IRQ, RESET, F*n*). If the FCCU does not react within a predefined interval as specified in the “FCCU configuration” section of the “Device configuration” chapter, on one of those outputs to the incoming failure indication, the FOSU assumes the FCCU has failed, and causes a destructive reset.

The FOSU does not require any configuration by software.

6.4.2 Common Cause Failure measures

Various measures are included to prevent CCFs from endangering the effectiveness of the safety measures. These measures include physical separation of the components on the die, routing restrictions and supervision of clock, power, temperature, test and debug signals. In general these measures are independent from the software.

There are also configuration registers that, on change, can affect the execution of the MCU's safety function and, simultaneously, disable the respective safety mechanism. These are either protected against bit flips or those flips are detected by independent measures. These same registers are also protected against accidental software writes by employing the register protection described above.

6.4.3 ECC

Error correcting codes are used for end-to-end protection from cores to system storage as well as for individual protection of peripheral RAMs.

6.4.3.1 ECC for storage

All storage used in normal operation is protected by ECC with:

- SEC/DED (single bit error correction and double bit error detection) for core, local memory, peripheral RAM
- SEC/DEC/TED (single bit error correction, double bit error correction and triple error bit detection) for Flash memory

The following exceptions exist:

- The caches in the cores are protected by EDC (Error Detection Code), only detecting, but not correcting errors.
- SRAM is protected by SEC/DEC ECC in PRAM controller only during partial writes (RMW operations) otherwise it is protected by E2E-ECC mechanism
- The HSM's AES program extension RAM are not protected by ECC.

A list showing the implementation of RAMs with ECC (including address protection) and for the SPC584Bx is shown in *Table 15*.

Table 15. ECC RAM implementations

Location	Memory	MEMU Classification	ECC	Address is ECC	Muxing Factor	Reacts to uncorrectable error
Core_2	D-MEM	System RAM	SEC/DED(1)	Y	16	Y
	I-Cache	System RAM	EDC	Y	4	Y
	D-Cache	System RAM	EDC	Y	4	Y
	ITAG	System RAM	SEC/DED	Y	4	Y
	DTAG	System RAM	SEC/DED	Y	4	Y
HSM	PRAM	Peripheral	SEC/DED	Y	8	N
	I-Cache	Peripheral	SEC/DED	Y	4	N
	ITAG	Peripheral	SEC/DED	Y	4	N
	C3	None	NO	N	4	N
Ethernet	FIFO RX	Peripheral	SEC/DED	Y	4	Y
	FIFO TX	Peripheral	SEC/DED	Y	4	Y
CAN	CAN	Peripheral	SEC/DED	N	8	Y
DMA	DMA	Peripheral	SEC/DED	Y	4	Y
Platform	SRAM	System RAM	E2E-ECC(2)	Y	8	Y
	Nexus	Peripheral	NO	—	4	N

1. Only when accessed by its own core. During access from other cores it is protected by E2E-ECC.
2. SEC/DEC for partial write operation (RMW), refer to *Section 34.5.13: Safety considerations*.

As shown in *Table 15*, some memories, particularly system storage, use an ECC computed over data and address to detect data and addressing faults (no/wrong/multiple selection). In addition, these same memories include dedicated measures against addressing and control faults (such as address/control feedback). This is different for storage related to PeMos. PeMos, in general, use an ECC without address error protections and do not include additional measures to detect them. All ECC calculations for RAMs in PeMos are accomplished by logic outside of the RAM, not in the RAM block.

6.4.3.2 All-x words and ECC

There is a special case for valid ECC values in the SPC584Bx. Memory entries that are all zeros (All-0) or all ones (All-1), including the ECC parity bits, are not valid for memory that is checked by ECC. Memories that include addresses in the ECC calculation do not specifically protect against All-0 or All-1. This means that for some addresses All-0 or All-1 may be valid. The Flash memory allows All-1, corresponding to the status of an erase block, as a valid codeword.

Memories that include addresses in the ECC calculation do not specifically protect against All-0 or All-1. This means that for some addresses All-0 or All-1 may be valid.

In case of not valid codeword, All-0 and All-1 memory content is indicated in different ways. For memories that do not include address into the ECC calculation, All-0 and All-1 will be always uncorrectable errors. For all memories that include address into the ECC code-bit calculation, since the ECC checkbits depend on the address, it is not possible to generate an uncorrectable error indication for all the possible addresses. Therefore, a not valid All-x codeword may result in a correctable error.

Note: For Flash memory, additional dedicated safety mechanisms exist to detect failures that have the potential of leading to an All-1 word (refer to Chapter 35: Embedded Flash Memory, for more details on Flash memory safety mechanisms).

6.4.3.3 ECC failure handling

Single-bit and double-bit errors (correctable and uncorrectable errors) are signaled to the FCCU unless filtered by the MEMU. The MEMU (refer to *Section 6.4.3.4: Memory Error Management Unit (MEMU)*) may filter ECC error notification for known ECC error addresses with same (optional) syndrome (known permanent correctable errors). Actual implementation will signal errors, not to the FCCU, but to the MEMU, which filters and then forwards unfiltered notifications in an aggregated manner to the FCCU.

6.4.3.4 Memory Error Management Unit (MEMU)

The MEMU is responsible for the collection and reporting of error events generated by the ECC logic used to protect system RAMs (in this context they are those RAMs with “System RAM” memory classification as shown in *Table 15*), peripheral RAM, and Flash memory. In addition to storage errors, e2eECC errors caused by XBAR, RAM or flash memory controllers and memory related faults detected during MBIST are also routed to MEMU. When ECC error events occur, the MEMU receives an error signal that causes an event to be recorded, and possibly sets corresponding error flags that are reported to the FCCU.

Refer to the Device Configuration chapter for MEMU connections.

The MEMU stores the addresses of ECC errors that have occurred in a table as follows:

- Uniqueness of the addresses with same (optional) syndrome in the table is ensured. For example, if an address with same (optional) syndrome is already stored in the

MEMU's table, correctable errors at that address will no longer be reported to the FCCU

- Software can read and modify the MEMU table and remove individual entries (by marking them as invalid). For example, this allows software to invalidate a new entry in the MEMU table and wait for a repeat occurrence of the ECC error indicating a permanent error in memory
- If the MEMU table overflows, an additional signal (instead of the ECC error signal) is sent to the FCCU

As the MEMU aggregates address information from several sources, it might not be able to process all simultaneously arriving reports. This is called a simultaneous overflow and is signaled to the FCCU as a buffer overflow (refer to *Chapter 73: Memory Error Management Unit (MEMU)* for more information).

6.4.3.4.1 Interface to ECC units

The MEMU receives data according to the following interface per ECC unit connected:

- Whether the error is a Single-Bit-Error (SBE) or Uncorrectable Error (UCE) type
- Address of the memory where the error occurred
- The (optional) ECC-syndrome or MBIST bad-bit information in case of SBE
- A configuration specifying whether the ECC unit is attached to a peripheral RAM, to a System RAM or to Flash memory

Not all ECC units provide a syndrome. If an ECC unit does not provide the syndrome, 0xFF is stored in the MEMU error table instead of the syndrome. If an error is signaled, it is compared against all errors known for that storage:

- If no entry of that address with same (optional) syndrome is already in the buffer, it is to be added into the appropriate table depending on the SBE versus UCE indicator
- If there is no free entry left in that buffer, the overflow flag is set

Note: If a previously known error has been marked invalid by software, no comparison against the address will occur.

If a valid entry of that address is in the buffer, and if the entry indicates an SBE in that address, and the error indicated is uncorrectable, a new entry is added in the MEMU buffer. In case of the same error is reported by ECC which reports the syndrome and additionally by ECC which does not, the reporting order becomes important.

- If the entry indicates an SBE in that address, and the error indicated is uncorrectable, a new entry is added in the MEMU buffer. If ECC which provides the syndrome reports first, the second reporting without syndrome is discarded.
- If ECC which does not provide the syndrome reports first, a second entry with the syndrome is stored in the table.

7 Device configuration

7.1 Introduction

This chapter provides details on the individual modules of the microcontroller. It includes:

- Module block diagrams showing immediate connections within the device
- Specific module-to-module interactions not necessarily discussed in the individual Module chapters
- Links for more information

7.2 Core modules

The HSM module embeds one additional core.

The main platform has one core that performs various computational and control functions: Main Core_2 which uses an e200z420n3 core.

PowerISA 2.06 and implementation-specific SPRs for the e200z215An3 core are listed in *Table 16*. All registers are 32 bits in size. Register bits are numbered from bit 0 to bit 31 (most-significant to least-significant). Shaded entries represent optional registers. An SPR register may be read or written with the **mfspr** and **mtspr** instructions. In the instruction syntax, compilers should recognize the mnemonic name given in *Table 16*.

Table 16. Special Purpose Registers (SPRs)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
XER	Integer Exception Register	1	R/W	No	No
LR	Link Register	8	R/W	No	No
CTR	Count Register	9	R/W	No	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No
SRR1	Save/Restore Register 1	27	R/W	Yes	No
PID0	Process ID Register	48	R/W	Yes	No
CSRR0	Critical Save/Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/Restore Register 1	59	R/W	Yes	No
DEAR	Data Exception Address Register	61	R/W	Yes	No
ESR	Exception Syndrome Register	62	R/W	Yes	No
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
SPRG0	SPR General 0	272	R/W	Yes	No
SPRG1	SPR General 1	273	R/W	Yes	No
SPRG2	SPR General 2	274	R/W	Yes	No
SPRG3	SPR General 3	275	R/W	Yes	No
PIR	Processor ID Register	286	R/W	Yes	No

Table 16. Special Purpose Registers (SPRs) (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
PVR	Processor Version Register	287	Read-only	Yes	No
DBSR	Debug Status Register	304	Read/Clear (1)	Yes	No
DBCR0	Debug Control Register 0	308	R/W	Yes	No
DBCR1	Debug Control Register 1	309	R/W	Yes	No
DBCR2	Debug Control Register 2	310	R/W	Yes	No
IAC1	Instruction Address Compare 1	312	R/W	Yes	No
IAC2	Instruction Address Compare 2	313	R/W	Yes	No
IAC3	Instruction Address Compare 3	314	R/W	Yes	No
IAC4	Instruction Address Compare 4	315	R/W	Yes	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	Yes	No
DVC1	Data Value Compare 1(2)	318	R/W	Yes	No
DVC2	Data Value Compare 2(2)	319	R/W	Yes	No
SPEFSCR	LSP/EFP APU status and control register	512	R/W	No	No
L1CFG0	L1 cache config register 0	515	Read-only	No	Yes
L1CFG1	L1 cache config register 1	516	Read-only	No	Yes
NPIDR	Nexus 3 Process ID register	517	R/W	No	Yes
DBCR3	Debug control register 3	561	R/W	Yes	Yes
DBCNT	Debug Counter register	562	R/W	Yes	Yes
DBCR4	Debug control register 4	563	R/W	Yes	Yes
DBCR5	Debug control register 5	564	R/W	Yes	Yes
IAC5	Instruction Address Compare 5	565	R/W	Yes	Yes
IAC6	Instruction Address Compare 6	566	R/W	Yes	Yes
IAC7	Instruction Address Compare 7	567	R/W	Yes	Yes
IAC8	Instruction Address Compare 8	568	R/W	Yes	Yes
MCSRR0	Machine Check Save/Restore Register 0	570	R/W	Yes	Yes
MCSRR1	Machine Check Save/Restore Register 1	571	R/W	Yes	Yes
MCSR	Machine Check Syndrome Register	572	Read/Clear (3)	Yes	Yes
MCAR	Machine Check Address Register	573	R/W	Yes	Yes
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
DDAM	Debug Data Acquisition Messaging register	576	R/W	No	Yes
DAC3	Data Address Compare 3	592	R/W	Yes	Yes

Table 16. Special Purpose Registers (SPRs) (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
DAC4	Data Address Compare 4	593	R/W	Yes	Yes
DBCR7	Debug control register 7	596	R/W	Yes	Yes
DBCR8	Debug control register 8	597	R/W	Yes	Yes
DDEAR	Debug Data Effective Address register	600	R/W	Yes	Yes
DVC1U(4)	Data Value Compare 1 Upper	601	R/W	Yes	No
DVC2U(4)	Data Value Compare 2 Upper	602	R/W	Yes	No
DBCR6	Debug control register 6	603	R/W	Yes	Yes
MAS0	MPU assist register 0	624	R/W	Yes	Yes
MAS1	MPU assist register 1	625	R/W	Yes	Yes
MAS2	MPU assist register 2	626	R/W	Yes	Yes
MAS3	MPU assist register 3	627	R/W	Yes	Yes
EDBRAC0	External debug resource allocation control register 0	638	Read only	Yes	Yes
MPU0CFG	MPU0 configuration register	692	Read-only	Yes	Yes
L1FINV1	L1 cache flush and invalidate control register 0	959	R/W	Yes	Yes
DEVENT	Debug Event register	975	R/W	No	Yes
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
L1CSR0	L1 cache control and status register 0	1010	R/W	Yes	Yes
L1CSR1	L1 cache control and status register 1	1011	R/W	Yes	Yes
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes
MPU0CSR0	MPU0 configuration register	1014	R/W	Yes	Yes
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
L1FINV0	L1 cache flush and invalidate control register 0	1016	R/W	Yes	Yes
SVR	System Version Register	1023	Read-only	Yes	Yes

1. The Debug Status Register can be read using *mfspr RT,DBSR*. The Debug Status Register cannot be directly written to. Instead, bits in the Debug Status Register corresponding to ‘1’ bits in GPR(RS) can be cleared using *mtspr DBSR,RS*.
2. Undefined on **POR** assertion, unchanged on **external PORESET** assertion.
3. The Machine Check Syndrome Register can be read using *mfspr RT,MCSR*. The Machine Check Syndrome Register cannot be directly written to. Instead, bits in the Machine Check Syndrome Register corresponding to ‘1’ bits in GPR(RS) can be cleared using *mtspr MCSR,RS*.
4. The 64-bit DVC registers are accessed by using the DVC1,2 spr #s for the lower 32 bits, and the DVC1,2U spr #s for the upper 32 bits.

Table 17 shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately following a system reset.

Table 17. Reset settings for e200z4 resources

Resource	System reset setting
Program Counter	p_rstbase[0:29] || 2'b00(1)
GPRs	Unaffected(2)
CR	Unaffected(2)
BUCSR	0x0000_0000
CSRR0	Unaffected(2)
CSRR1	Unaffected(2)
CTR	Unaffected(2)
DAC1	0x0000_0000(3)
DAC2	0x0000_0000(3)
DAC3	0x0000_0000(3)
DAC4	0x0000_0000(3)
DBCR0	0x0000_0000(3)
DBCR1	0x0000_0000(3)
DBCR2	0x0000_0000(3)
DBCR4	0x0000_0000(3)
DBCR5	0x0000_0000(3)
DBCR6	0x0000_0000(3)
DBCR7	0x0000_0000(3)
DBCR8	0x0000_0000(3)
DBSR	0x1000_0000[3]
DDAM	0x0000_0000(3)
DDEAR	Unaffected(2)
DEAR	Unaffected(2)
DEVENT	0x0000_0000(3)
DSRR0	Unaffected(2)
DSRR1	Unaffected(2)
DVC1	Unaffected(2)
DVC2	Unaffected(2)
ESR	0x0000_0000
HID0	0x0000_0000
HID1	0x0000_0000
IAC1	0x0000_0000(3)
IAC2	0x0000_0000(3)
IAC3	0x0000_0000(3)
IAC4	0x0000_0000(3)

Table 17. Reset settings for e200z4 resources (continued)

Resource	System reset setting
IAC5	0x0000_0000(3)
IAC6	0x0000_0000(3)
IAC7	0x0000_0000(3)
IAC8	0x0000_0000(3)
IVPR	Unaffected(2)
LR	Unaffected(2)
L1CFG0, L1CFG1(4)	—
L1CSR0, 1	0x0000_0000
L1FINV0, 1	0x0000_0000
MAS1	Unaffected(2)
MAS2	Unaffected(2)
MAS3	Unaffected(2)
MAS4	Unaffected(2)
MCAR	Unaffected(2)
MCSR	0x0000_0000
MCSRR0	Unaffected(2)
MCSRR1	Unaffected(2)
MMUCFG(4)	—
MPU0CSR0	0x0000_0000
MPU0CFG(4)	—
MSR	0x0000_0000
NPIDR	0x0000_0000
PID0	0x0000_0000
PIR	0x0000_00 || p_cpuid[0:7]1
PVR(4)	—
SPEFSCR	0x0000_0000
SPRG0	Unaffected(2)
SPRG1	Unaffected(2)
SPRG2	Unaffected(2)
SPRG3	Unaffected(2)
SRR0	Unaffected(2)
SRR1	Unaffected(2)
SVR(4)	—
XER	0x0000_0000

1. The levels that determine these bits are set by the factory and may change between revisions of the device.
Main_Core_2: p_cpuid[0:7] = 0x02
2. Undefined on **POR** assertion, unchanged on **external PORESET** assertion.
3. Reset by processor reset **external PORESET** if DBCR0[EDM]=0, as well as unconditionally by **POR**.
4. Read-only registers.

Table 18. Core configuration inputs

CPU ID	Inputs	Value	Affected registers
Core_2	p_dmem_rstcfg[8:0]	0x0DA	DMEMCTL0
	p_e2e_rstcfg[3:0]	0x5	E2ECTL0

7.3 System modules

7.3.1 SIUL2 configuration

The System Integration Unit Lite 2 (SIUL2) controls the MCU pad configuration, ports, general-purpose input and output (GPIO) signals and external interrupts with trigger event configuration.

For details please refer *Chapter 16: System Integration Unit Lite2 (SIUL2)*.

Note: All output pins, whose SIUL2_MSCRn[SMC] = 0, disable their output drivers when the device enters SAFE mode while the internal weak pull-up of GPIO is enabled automatically. Coherently, ME_SAFE_MC.PDO must be set to 1 (default value) when disabling the output buffer.
Each other configuration has no effect on output driver of pin, refer to Table 19.

Table 19. Output buffer configuration

ME_SAFE_MC [PDO]	SIUL2.MSCR_IOn[SMC]	DRUN Mode	SAFE Mode
0 (no safe gating of I/O)	0 (output buffer disabled)	Output buffer enabled	Output buffer enabled
0 (no safe gating of I/O)	1 (output buffer enabled)	Output buffer enabled	Output buffer enabled
1 (output buffer driver disabled in SAFE mode)	0 (output buffer disabled)	Output buffer enabled	Output buffer disabled, internal weak pull-up enabled
1 (output buffer driver disabled in SAFE mode)	1 (output buffer enabled)	Output buffer enabled	Output buffer enabled

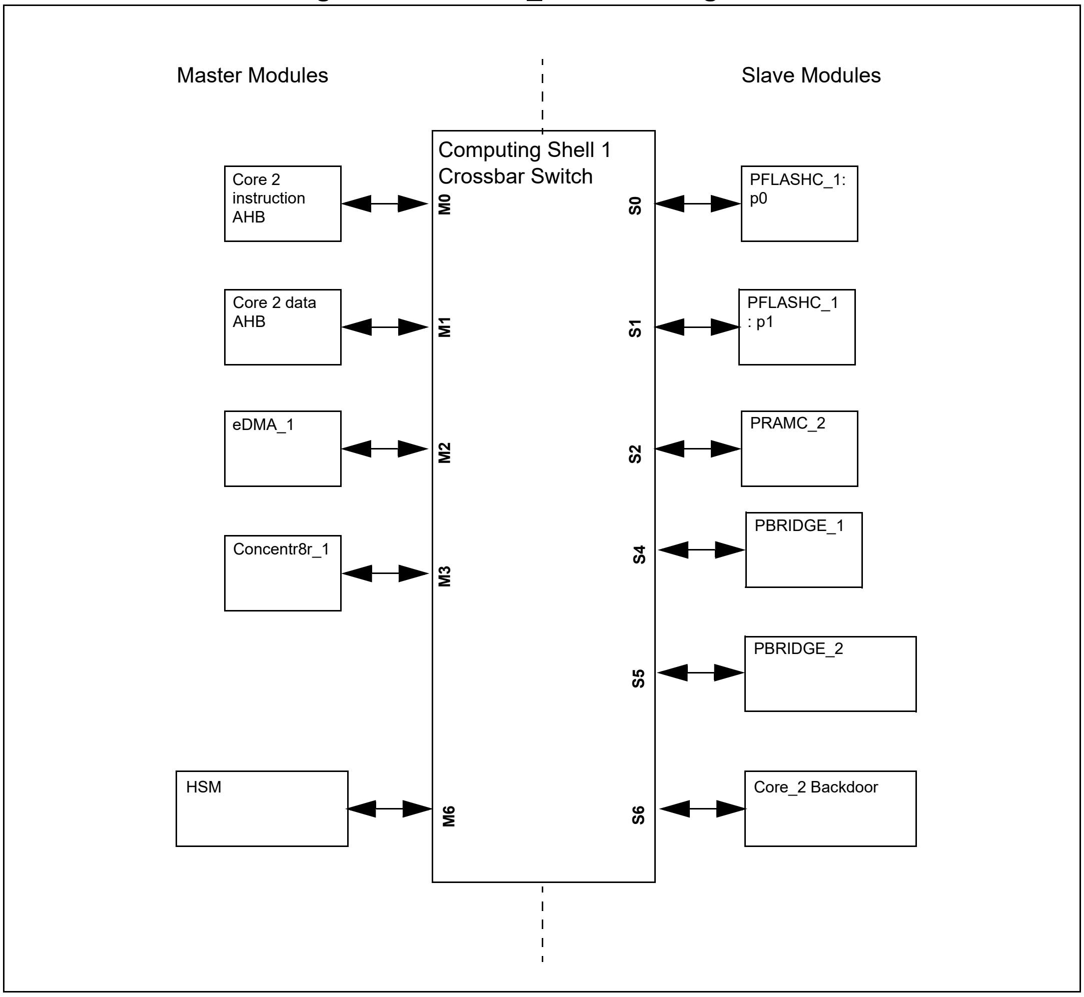
7.3.2 Crossbar switch

This device contains one crossbar switch module.

7.3.2.1 Computing shell 1 crossbar switch (XBAR_1) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, refer to the chapter for that module.

Figure 7. Crossbar_1 switch integration



The following figure shows hardwired connections between Masters and Flash Ports slaves (S0/S1).

Figure 8. Crossbar_1 hardwired connections

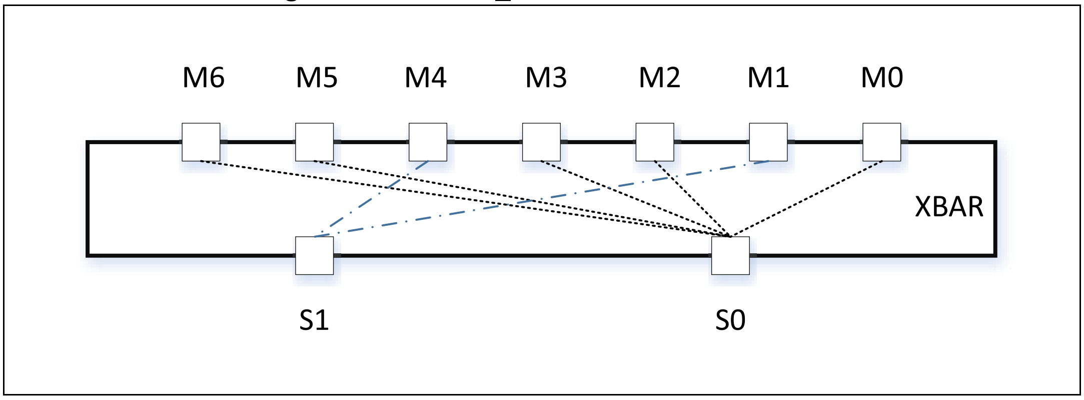


Table 20. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	*Chapter 17: Crossbar switch (XBAR)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 25: Clocking*
Crossbar switch master	Cores	*Section 7.2: Core modules*
Crossbar switch slave	Flash memory controller	*Section 3.3.1: Flash memory controller*
Crossbar switch slave	Processor core local memory	*Section 3.2.2: Processor core local RAM*
Crossbar switch slave	System RAM	*Section 3.2.1: System SRAM*

7.3.2.1.1 Crossbar_1 switch master assignments

This device contains six masters connected to the computing shell 1 crossbar switch.

The master assignment is shown in *Table 21*.

Table 21. Crossbar_1 switch master assignments

Master module	Physical master port number
Core_2 Instruction AHB	M0
Core_2 Data AHB	M1
eDMA_1	M2
Concentrator_1	M3
Reserved	M4
Reserved	M5
HSM	M6

7.3.2.1.2 Crossbar_1 switch slave assignments

This device contains eight slaves connected to the computing shell 1 crossbar switch.

The slave assignment is shown in *Table 22*.

Table 22. Crossbar_1 switch slave assignments

Slave module	Physical slave port number
PFLASHC_1:p0	S0
PFLASHC_1:p1	S1
PRAMC_2	S2
Reserved	S3
PBRIDGE_1	S4
PBRIDGE_2	S5

Table 22. Crossbar_1 switch slave assignments (continued)

Slave module	Physical slave port number
Core_2 Backdoor	S6
Reserved	S7

7.3.2.1.3 PRS register reset values

The computing shell 1 XBAR_1 PRS*n* registers reset to 0x55531024. Note that these registers are read-only.

7.3.2.1.4 MGPCR register reset values

The streaming shell 1 XBAR_1 MGPCRn registers reset to 0x00000000.

7.3.2.2 AHB Master ID Assignments

Table 23. SMPU1 logical bus master assignments

SMPU logical bus master number	Bus master
0	—
1	—
2	Core_2
3	—
4	Ethernet_0
5	—
6	—
7	SIPI
8	—
9	—
10	Core_2 Nexus
11	eDMA_1
12	—
13	—
14	HSM
15	—

7.3.3 System Memory Protection Unit (SMPU) configuration

This chip contains two instances of the System Memory Protection Unit (SMPU):

- SMPU_HSM for the Hardware Security Module
- SMPU for XBAR

The following sections explain how SMPU for XBAR is configured on this chip. See the *Microcontroller Security Reference Manual* for configuration details of SMPU_HSM.

7.3.3.1 SMPU1 configuration

This section summarizes how the module instance has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

Figure 9. SMPU1 configuration

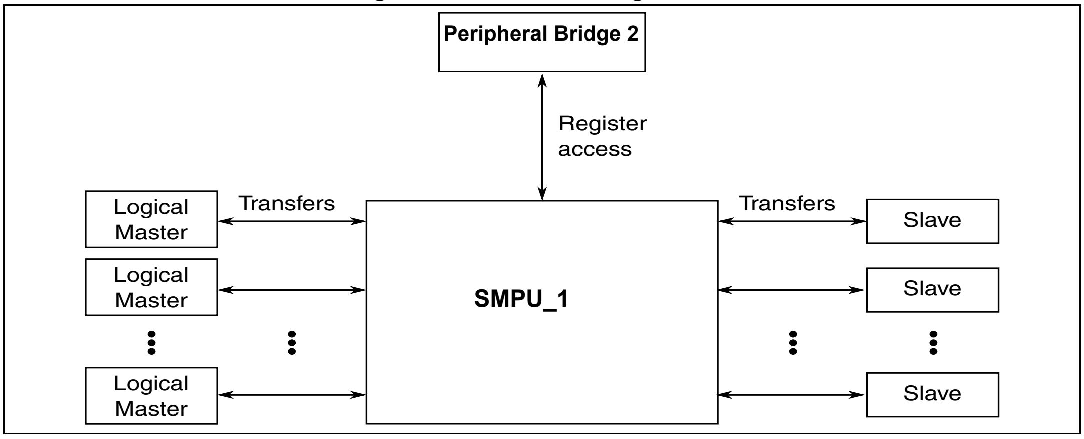


Table 24. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	*Chapter 20: System Memory Protection Unit (SMPU)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 25: Clocking*
Power management	—	*Chapter 10: Power management*
Logical masters	—	*Section 7.3.3.1.1: SMPU1 logical bus master assignments*
Region descriptors	—	*Section 7.3.3.1.2: Number of region descriptors supported by SMPU1*
RGD_WORD2 formats	—	*Section 7.3.3.1.3: RGD_WORD2 formats supported by SMPU1*
Master Access Permissions	—	*Section 7.3.3.1.4: Master Access permissions for SMPU1*

7.3.3.1.1 SMPU1 logical bus master assignments

The logical bus master assignments for the SMPU1 are present in Device configuration chapter, *Section 7.3.2.2: AHB Master ID Assignments*.

7.3.3.1.2 Number of region descriptors supported by SMPU1

This instance of the SMPU module supports a maximum of 16 region descriptors.

7.3.3.1.3 RGD_WORD2 formats supported by SMPU1

For RGD_WORD2, this instance of the SMPU module supports only the FMT0 format. The RGD_WORD3[FMT] field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.

7.3.3.1.4 Master Access permissions for SMPU1

The only masters allowed to access the SMPU1 registers are HSM, Core 2. It is also required that the core be in supervisor mode.

7.3.4 Peripheral bridge configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

Figure 10. Peripheral bridge configuration

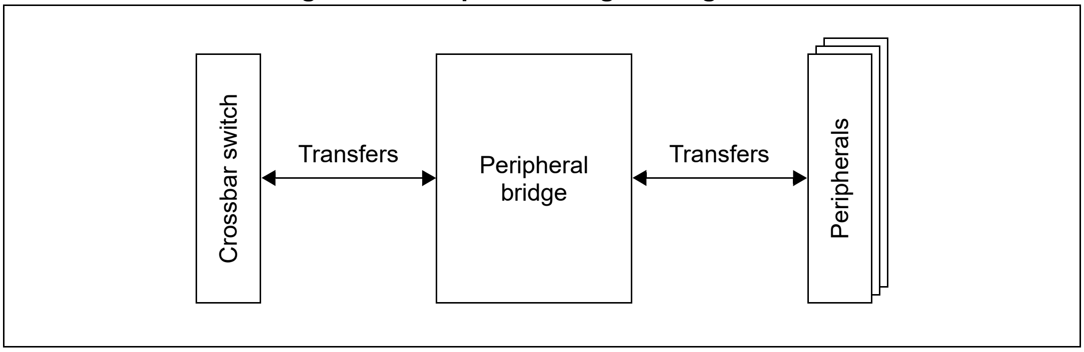


Table 25. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge	*Chapter 19: Peripheral Bridge (PBRIDGE)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 25: Clocking*
Crossbar switch	Crossbar switch	*Section 7.3.2: Crossbar switch*

7.3.4.1 Peripheral bridge instantiation

This device contains two identical peripheral bridges.

7.3.4.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See *Chapter 5: Memory Map* for the memory slot assignment for each module.

7.3.4.3 MPR registers

Each of the two peripheral bridges supports up to sixteen crossbar switch masters, each assigned to a MPROT*n* field in the MPRA-B registers. However, fewer are supported on this device. See *Section 7.3.2: Crossbar switch* for details of the master port assignments for this device.

7.3.4.4 PACR/OPACR registers

The peripherals attached to the peripheral bridges each assigned to a PACR*n* or OPACR*n* field within the PACRA–H/OPACRA–AF registers. However, fewer peripherals are supported on this device. See *Chapter 5: Memory Map* for details of the peripheral slot assignments for this device. Unused PACR*n* fields are reserved.

7.3.5 Interrupt controller (INTC) configuration

7.3.5.1 INTC configurable parameters

The INTC on this chip supports two processors.

7.3.5.2 Interrupt channel map

Table 26 defines the interrupt sources for the INTC on this chip. All IRQ numbers not specifically listed are not used.

Table 26. Interrupt sources

Number	Description	Register bit
0	Software interrupt flag 0	INTC_1_SSCIR0[CLR]
1	Software interrupt flag 1	INTC_1_SSCIR1[CLR]
2	Software interrupt flag 2	INTC_1_SSCIR2[CLR]
3	Software interrupt flag 3	INTC_1_SSCIR3[CLR]
4	Software interrupt flag 4	INTC_1_SSCIR4[CLR]
5	Software interrupt flag 5	INTC_1_SSCIR5[CLR]
6	Software interrupt flag 6	INTC_1_SSCIR6[CLR]
7	Software interrupt flag 7	INTC_1_SSCIR7[CLR]
8	Software interrupt flag 8	INTC_1_SSCIR8[CLR]
9	Software interrupt flag 9	INTC_1_SSCIR9[CLR]
10	Software interrupt flag 10	INTC_1_SSCIR10[CLR]
11	Software interrupt flag 11	INTC_1_SSCIR11[CLR]
12	Software interrupt flag 12	INTC_1_SSCIR12[CLR]
13	Software interrupt flag 13	INTC_1_SSCIR13[CLR]
14	Software interrupt flag 14	INTC_1_SSCIR14[CLR]

Table 26. Interrupt sources (continued)

Number	Description	Register bit
15	Software interrupt flag 15	INTC_1_SSCIR15[CLR]
16	Software interrupt flag 16	INTC_1_SSCIR16[CLR]
17	Software interrupt flag 17	INTC_1_SSCIR17[CLR]
18	Software interrupt flag 18	INTC_1_SSCIR18[CLR]
19	Software interrupt flag 19	INTC_1_SSCIR19[CLR]
20	Software interrupt flag 20	INTC_1_SSCIR20[CLR]
21	Software interrupt flag 21	INTC_1_SSCIR21[CLR]
22	Software interrupt flag 22	INTC_1_SSCIR22[CLR]
23	Software interrupt flag 23	INTC_1_SSCIR23[CLR]
24	Software interrupt flag 24	INTC_1_SSCIR24[CLR]
25	Software interrupt flag 25	INTC_1_SSCIR25[CLR]
26	Software interrupt flag 26	INTC_1_SSCIR26[CLR]
27	Software interrupt flag 27	INTC_1_SSCIR27[CLR]
28	Software interrupt flag 28	INTC_1_SSCIR28[CLR]
29	Software interrupt flag 29	INTC_1_SSCIR29[CLR]
30	Software interrupt flag 30	INTC_1_SSCIR30[CLR]
31	Software interrupt flag 31	INTC_1_SSCIR31[CLR]
34	Watchdog timer 2 (SWT_2)	SWT_2_IR[TIF]
35	Watchdog timer 3 (SWT_3)	SWT_3_IR[TIF]
44	Platform periodic timer 2 (STM_2)	STM_2_CIR0[CIF]
45		STM_2_CIR1[CIF]
46		STM_2_CIR2[CIF]
47		STM_2_CIR3[CIF]
52	eDMA Combined Channel Error Flags 0 - 63	DMA_1_ERRL[INT0 - INT31] :OR: DMA_1_ERRH[INT32 - INT63]

Table 26. Interrupt sources (continued)

Number	Description	Register bit
53	Enhanced Direct Memory Access 1 (eDMA_1)	DMA_1_INTL[INT0]
54		DMA_1_INTL[INT1]
55		DMA_1_INTL[INT2]
56		DMA_1_INTL[INT3]
57		DMA_1_INTL[INT4]
58		DMA_1_INTL[INT5]
59		DMA_1_INTL[INT6]
60		DMA_1_INTL[INT7]
61		DMA_1_INTL[INT8]
62		DMA_1_INTL[INT9]
63		DMA_1_INTL[INT10]
64		DMA_1_INTL[INT11]
65		DMA_1_INTL[INT12]
66		DMA_1_INTL[INT13]
67		DMA_1_INTL[INT14]
68		DMA_1_INTL[INT15]
69		DMA_1_INTL[INT16]
70		DMA_1_INTL[INT17]
71		DMA_1_INTL[INT18]
72		DMA_1_INTL[INT19]
73		DMA_1_INTL[INT20]
74		DMA_1_INTL[INT21]
75		DMA_1_INTL[INT22]
76		DMA_1_INTL[INT23]
77		DMA_1_INTL[INT24]
78		DMA_1_INTL[INT25]
79		DMA_1_INTL[INT26]
80		DMA_1_INTL[INT27]
81		DMA_1_INTL[INT28]
82		DMA_1_INTL[INT29]
83		DMA_1_INTL[INT30]
84		DMA_1_INTL[INT31]
185	Flash controller Prog/Erase/Suspend IRQ	MCR[DONE]

Table 26. Interrupt sources (continued)

Number	Description	Register bit
212	Ethernet 0 (SYNOPSYS IP)	ETHERNET_0[sbd_intr_o] combined interrupts from all EQOS sources
213		ETHERNET_0[pmt_intr_o]
214		ETHERNET_0[lpi_intr_o]
224	Real Time Counter (RTC/API)	RTC
225		API wakeup
226	Periodic Interrupt Timer 0 (PIT_0)	PIT_0_TFLG0[TIF]
227		PIT_0_TFLG1[TIF]
228		PIT_0_TFLG2[TIF]
229		PIT_0_TFLG3[TIF]
230		PIT_0_TFLG4[TIF]
231		PIT_0_TFLG5[TIF]
232		PIT_0_TFLG6[TIF]
233		PIT_0_TFLG7[TIF]
239	Periodic Interrupt Timer 0 (PIT_0)	PIT_0_RTI_TFLG[TIF]
240	Periodic Interrupt Timer 1 (PIT_1)	PIT_1_TFLG0[TIF]
241		PIT_1_TFLG1[TIF]
242	XOSC counter	XOSC_CTL[I_OSC]
243	SIUL External Interrupt 0-7	SIUL2_DISR0[EIF0-EIF7] (EXT_INT 0 signal)
244	SIUL External Interrupt 8-15	SIUL2_DISR0[EIF8-EIF15] (EXT_INT 1 signal)
245	SIUL External Interrupt 16-19	SIUL2_DISR0[EIF16-EIF19] (EXT_INT 2 signal)
246	SIUL External Interrupt 20-23	SIUL2_DISR0[EIF20-EIF23] (EXT_INT 3 signal)
247	WakeUp unit (WKPU)	WakeUp_IRQ_0
248		WakeUp_IRQ_1
249		WakeUp_IRQ_2
250		WakeUp_IRQ_3
251	Magic Carpet Module - ME 0	ME_IS[I_SAFE]
252	Magic Carpet Module - ME 1	ME_IS[I_MTC]
253	Magic Carpet Module - ME 2	ME_IS[I_IMODE]
254	Magic Carpet Module - ME 3	ME_IS[I_ICONF]
255	Magic Carpet Module - RGM 0	MC_RGM Functional and destructive reset alternate event interrupt

Table 26. Interrupt sources (continued)

Number	Description	Register bit
259	Serial Peripheral Interface 0 (DSPI_0)	DSPI_0_SR[TFUF] :OR: DSPI_0_SR[RFOF] :OR: DSPI_0_SR[TFIWF]
260		DSPI_0_SR[EOQF]
261		DSPI_0_SR[TFFF]
262		DSPI_0_SR[TCF]
263		DSPI_0_SR[RFDF]
264		DSPI_0_SR[CMD_TCF]
265		DSPI_0_SR[CMDFFF]
266		DSPI_0_SR[SPEF]
268	Serial Peripheral Interface 1 (DSPI_1)	DSPI_1_SR[TFUF] :OR: DSPI_1_SR[RFOF] :OR: DSPI_1_SR[TFIWF]
269		DSPI_1_SR[EOQF]
270		DSPI_1_SR[TFFF]
271		DSPI_1_SR[TCF]
272		DSPI_1_SR[RFDF]
273		DSPI_1_SR[CMD_TCF]
274		DSPI_1_SR[CMDFFF]
275		DSPI_1_SR[SPEF]
277	Serial Peripheral Interface 2 (DSPI_2)	DSPI_2_SR[TFUF] :OR: DSPI_2_SR[RFOF] :OR: DSPI_2_SR[TFIWF]
278		DSPI_2_SR[EOQF]
279		DSPI_2_SR[TFFF]
280		DSPI_2_SR[TCF]
281		DSPI_2_SR[RFDF]
282		DSPI_2_SR[CMD_TCF]
283		DSPI_2_SR[CMDFFF]
284		DSPI_2_SR[SPEF]

Table 26. Interrupt sources (continued)

Number	Description	Register bit
286	Serial Peripheral Interface 3 (DSPI_3)	DSPI_3_SR[TFUF] :OR: DSPI_3_SR[RFOF] :OR: DSPI_3_SR[TFIWF]
287		DSPI_3_SR[EOQF]
288		DSPI_3_SR[TFFF]
289		DSPI_3_SR[TCF]
290		DSPI_3_SR[RFDF]
291		DSPI_3_SR[CMD_TCF]
292		DSPI_3_SR[CMDFFF]
293		DSPI_3_SR[SPEF]
295	Serial Peripheral Interface 4 (DSPI_4)	DSPI_4_SR[TFUF] :OR: DSPI_4_SR[RFOF] :OR: DSPI_4_SR[TFIWF]
296		DSPI_4_SR[EOQF]
297		DSPI_4_SR[TFFF]
298		DSPI_4_SR[TCF]
299		DSPI_4_SR[RFDF]
300		DSPI_4_SR[CMD_TCF]
301		DSPI_4_SR[CMDFFF]
302		DSPI_4_SR[SPEF] :OR: DSPI_4_SR[DPEF]
303		DSPI_4_SR[DDIF]
304	Serial Peripheral Interface 5 (DSPI_5)	DSPI_5_SR[TFUF] :OR: DSPI_5_SR[RFOF] :OR: DSPI_5_SR[TFIWF]
305		DSPI_5_SR[EOQF]
306		DSPI_5_SR[TFFF]
307		DSPI_5_SR[TCF]
308		DSPI_5_SR[RFDF]
309		DSPI_5_SR[CMD_TCF]
310		DSPI_5_SR[CMDFFF]
311		DSPI_5_SR[SPEF] :OR: DSPI_4_SR[DPEF]
312		DSPI_5_SR[DDIF]

Table 26. Interrupt sources (continued)

Number	Description	Register bit
313	Serial Peripheral Interface 6 (DSPI_6)	DSPI_6_SR[TFUF] :OR: DSPI_6_SR[RFOF] :OR: DSPI_6_SR[TFIWF]
314		DSPI_6_SR[EOQF]
315		DSPI_6_SR[TFFF]
316		DSPI_6_SR[TCF]
317		DSPI_6_SR[RFDF]
318		DSPI_6_SR[CMD_TCF]
319		DSPI_6_SR[CMDFFF]
320		DSPI_6_SR[SPEF]
376	UART/LIN 0 (LINFlexD_0)	LINFlexD_0_RX
377		LINFlexD_0_TX
378		LINFlexD_0_ERR
380	UART/LIN 1 (LINFlexD_1)	LINFlexD_1_RX
381		LINFlexD_1_TX
382		LINFlexD_1_ERR
384	UART/LIN 2 (LINFlexD_2)	LINFlexD_2_RX
385		LINFlexD_2_TXI
386		LINFlexD_2_ERR
388	UART/LIN 3 (LINFlexD_3)	LINFlexD_3_RX
389		LINFlexD_3_TX
390		LINFlexD_3_ERR
392	UART/LIN 4 (LINFlexD_4)	LINFlexD_4_RX
393		LINFlexD_4_TX
394		LINFlexD_4_ERR
396	UART/LIN 5 (LINFlexD_5)	LINFlexD_5_RX
397		LINFlexD_5_TX
398		LINFlexD_5_ERR
400	UART/LIN 6 (LINFlexD_6)	LINFlexD_6_RX
401		LINFlexD_6_TX
402		LINFlexD_6_ERR
404	UART/LIN 7 (LINFlexD_7)	LINFlexD_7_RX
405		LINFlexD_7_TX
406		LINFlexD_7_ERR

Table 26. Interrupt sources (continued)

Number	Description	Register bit
408	UART/LIN 8 (LINFlexD_8)	LINFlexD_8_RX
409		LINFlexD_8_TX
410		LINFlexD_8_ERR
412	UART/LIN 9 (LINFlexD_9)	LINFlexD_9_RX
413		LINFlexD_9_TX
414		LINFlexD_9_ERR
416	UART/LIN 10 (LINFlexD_10)	LINFlexD_10_RX
417		LINFlexD_10_TX
418		LINFlexD_10_ERR
420	UART/LIN 11 (LINFlexD_11)	LINFlexD_11_RX
421		LINFlexD_11_TX
422		LINFlexD_11_ERR
424	UART/LIN 12 (LINFlexD_12)	LINFlexD_12_RX
425		LINFlexD_12_TXI
426		LINFlexD_12_ERR
436	UART/LIN 15 (LINFlexD_15)	LINFlexD_15_RX
437		LINFlexD_15_TX
438		LINFlexD_15_ERR
448	I2C_0	I2C0_SR[IBAL] :OR: I2C0_SR[TCF] :OR: I2C0_SR[IAAS]
477	Power Monitor Unit	OR-ed GR_S voltage detector flags
478	Power management Unit (temperature sensor)	OR-ed EPR_TD temperature detector flags
479	XOSC32k counter	counter expired
488	FCCU	FCCU_IRQ_STAT[ALRM_STAT]
489		FCCU_IRQ_STAT[CFG_TO_STAT]
495	STCU	STCU_RUNSW[MBIE]

Table 26. Interrupt sources (continued)

Number	Description	Register bit
496	HSM/CSM	HSM2HTIE[0]
497		HSM2HTIE[1]
498		HSM2HTIE[2]
499		HSM2HTIE[3]
500		HSM2HTIE[4]
501		HSM2HTIE[5]
502		HSM2HTIE[6]
503		HSM2HTIE[7]
504		HSM2HTIE[8]
505		HSM2HTIE[9]
506		HSM2HTIE[10]
507		HSM2HTIE[11]
508		HSM2HTIE[12]
509		HSM2HTIE[13]
510		HSM2HTIE[14]
511		HSM2HTIE[15]
512		HSM2HTIE[16]
513		HSM2HTIE[17]
514		HSM2HTIE[18]
515		HSM2HTIE[19]
516		HSM2HTIE[20]
517		HSM2HTIE[21]
518		HSM2HTIE[22]
519		HSM2HTIE[23]
520		HSM2HTIE[24]
521		HSM2HTIE[25]
522		HSM2HTIE[26]
523		HSM2HTIE[27]
524		HSM2HTIE[28]
525		HSM2HTIE[29]
526		HSM2HTIE[30]
527		HSM2HTIE[31]

Table 26. Interrupt sources (continued)

Number	Description	Register bit
528	SAR_ADC_12bit_0	combined interrupt signal: SARADC12_0 [NEOC] :OR: SARADC12_0 [NECH] :OR: SARADC12_0 [JEOC] :OR: SARADC12_0 [JECH]:OR: SARADC12_0 [WDGxL] :OR: SARADC12_0 [WDGxH]
529	SAR_ADC_12bit_1	combined interrupt signal: SARADC12_1 [NEOC] :OR: SARADC12_1 [NECH] :OR: SARADC12_1 [JEOC] :OR: SARADC12_1 [JECH]:OR: SARADC12_1 [WDGxL] :OR: SARADC12_1 [WDGxH]
543	SAR_ADC_12bit_SUPERVISOR	combined interrupt signal: SARADC12_SV [NEOC] :OR: SARADC12_SV [NECH] :OR: SARADC12_SV [JEOC] :OR: SARADC12_SV [JECH]:OR: SARADC12_SV [EOCTU] :OR: SARADC12_SV [WDGxL] :OR: SARADC12_SV [WDGxH]
557	SAR_ADC_10bit_STDBY	combined interrupt signal: SARADC10_SB [NEOC] :OR: SARADC10_SB [NECH] :OR: SARADC10_SB [JEOC] :OR: SARADC10_SB [JECH]:OR: SARADC10_SB [EOCTU] :OR: SARADC10_SB [WDGxL] :OR: SARADC10_SB [WDGxH]
674	JTAGM	JTAGM_SR[SPU_INT] :OR: JTAGM_SR[IDLE]
675	JDC	JDC_MSR[JIN_INT] :OR: JDC_MSR[JOUT_INT]
686	CAN_SUB_0_M_CAN_0	Line 0 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_0_IR[0] & ~M_CAN_0_ILS[0]):OR: ... :OR: (M_CAN_0_IR[11] & ~M_CAN_0_ILS[11]) :OR: (M_CAN_0_IR[13] & ~M_CAN_0_ILS[13]) :OR: … :OR: (M_CAN_0_IR[31] & ~M_CAN_0_ILS[31])
687		Line 1 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_0_IR[0] & M_CAN_0_ILS[0]):OR: ... :OR: (M_CAN_0_IR[11] & M_CAN_0_ILS[11]) :OR: (M_CAN_0_IR[13] & M_CAN_0_ILS[13]) :OR: … :OR: (M_CAN_0_IR[31] & M_CAN_0_ILS[31])

Table 26. Interrupt sources (continued)

Number	Description	Register bit
688	CAN_SUB_0_M_CAN_1	Line 0 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_1_IR[0] & ~M_CAN_1_ILS[0]):OR: ... :OR: (M_CAN_1_IR[11] & ~M_CAN_1_ILS[11]) :OR: (M_CAN_1_IR[13] & ~M_CAN_1_ILS[13]) :OR: … :OR: (M_CAN_1_IR[31] & ~M_CAN_1_ILS[31])
689	CAN_SUB_0_M_CAN_1	Line 1 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_1_IR[0] & M_CAN_1_ILS[0]):OR: ... :OR: (M_CAN_1_IR[11] & M_CAN_1_ILS[11]) :OR: (M_CAN_1_IR[13] & M_CAN_1_ILS[13]) :OR: … :OR: (M_CAN_1_IR[31] & M_CAN_1_ILS[31])
690	CAN_SUB_0_M_CAN_2	Line 0 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_2_IR[0] & ~M_CAN_2_ILS[0]):OR: ... :OR: (M_CAN_2_IR[11] & ~M_CAN_2_ILS[11]) :OR: (M_CAN_2_IR[13] & ~M_CAN_2_ILS[13]) :OR: … :OR: (M_CAN_2_IR[31] & ~M_CAN_2_ILS[31])
691	CAN_SUB_0_M_CAN_2	Line 1 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_2_IR[0] & M_CAN_2_ILS[0]):OR: ... :OR: (M_CAN_2_IR[11] & M_CAN_2_ILS[11]) :OR: (M_CAN_2_IR[13] & M_CAN_2_ILS[13]) :OR: … :OR: (M_CAN_2_IR[31] & M_CAN_2_ILS[31])
692	CAN_SUB_0_M_CAN_3	Line 0 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_3_IR[0] & ~M_CAN_3_ILS[0]):OR: ... :OR: (M_CAN_3_IR[11] & ~M_CAN_3_ILS[11]) :OR: (M_CAN_3_IR[13] & ~M_CAN_3_ILS[13]) :OR: … :OR: (M_CAN_3_IR[31] & ~M_CAN_3_ILS[31])
693	CAN_SUB_0_M_CAN_3	Line 1 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_3_IR[0] & M_CAN_3_ILS[0]):OR: ... :OR: (M_CAN_3_IR[11] & M_CAN_3_ILS[11]) :OR: (M_CAN_3_IR[13] & M_CAN_3_ILS[13]) :OR: … :OR: (M_CAN_3_IR[31] & M_CAN_3_ILS[31])

Table 26. Interrupt sources (continued)

Number	Description	Register bit
694	CAN_SUB_1_M_CAN_1	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_1_IR[0] & ~M_CAN_1_ILS[0]):OR: ... :OR: (M_CAN_1_IR[11] & ~M_CAN_1_ILS[11]) :OR: (M_CAN_1_IR[13] & ~M_CAN_1_ILS[13]) :OR: … :OR: (M_CAN_1_IR[31] & ~M_CAN_1_ILS[31])
695		Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_1_IR[0] & M_CAN_1_ILS[0]):OR: ... :OR: (M_CAN_1_IR[11] & M_CAN_1_ILS[11]) :OR: (M_CAN_1_IR[13] & M_CAN_1_ILS[13]) :OR: … :OR: (M_CAN_1_IR[31] & M_CAN_1_ILS[31])
696	CAN_SUB_1_M_CAN_2	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_2_IR[0] & ~M_CAN_2_ILS[0]):OR: ... :OR: (M_CAN_2_IR[11] & ~M_CAN_2_ILS[11]) :OR: (M_CAN_2_IR[13] & ~M_CAN_2_ILS[13]) :OR: … :OR: (M_CAN_2_IR[31] & ~M_CAN_2_ILS[31])
697		Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_2_IR[0] & M_CAN_2_ILS[0]):OR: ... :OR: (M_CAN_2_IR[11] & M_CAN_2_ILS[11]) :OR: (M_CAN_2_IR[13] & M_CAN_2_ILS[13]) :OR: … :OR: (M_CAN_2_IR[31] & M_CAN_2_ILS[31])
698	CAN_SUB_1_M_CAN_3	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_3_IR[0] & ~M_CAN_3_ILS[0]):OR: ... :OR: (M_CAN_3_IR[11] & ~M_CAN_3_ILS[11]) :OR: (M_CAN_3_IR[13] & ~M_CAN_3_ILS[13]) :OR: … :OR: (M_CAN_3_IR[31] & ~M_CAN_3_ILS[31])
699		Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_3_IR[0] & M_CAN_3_ILS[0]):OR: ... :OR: (M_CAN_3_IR[11] & M_CAN_3_ILS[11]) :OR: (M_CAN_3_IR[13] & M_CAN_3_ILS[13]) :OR: … :OR: (M_CAN_3_IR[31] & M_CAN_3_ILS[31])

Table 26. Interrupt sources (continued)

Number	Description	Register bit
700	CAN_SUB_1_M_CAN_4	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_4_IR[0] & ~M_CAN_4_ILS[0]):OR: ... :OR: (M_CAN_4_IR[11] & ~M_CAN_4_ILS[11]) :OR: (M_CAN_4_IR[13] & ~M_CAN_4_ILS[13]) :OR: … :OR: (M_CAN_4_IR[31] & ~M_CAN_4_ILS[31])
701		Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_4_IR[0] & M_CAN_4_ILS[0]):OR: ... :OR: (M_CAN_4_IR[11] & M_CAN_4_ILS[11]) :OR: (M_CAN_4_IR[13] & M_CAN_4_ILS[13]) :OR: … :OR: (M_CAN_4_IR[31] & M_CAN_4_ILS[31])
900	eMIOS_0	EMIOS_GFR[F0,F1]
901		EMIOS_GFR[F2,F3]
902		EMIOS_GFR[F4,F5]
903		EMIOS_GFR[F6,F7]
904		EMIOS_GFR[F8,F9]
905		EMIOS_GFR[F10,F11]
906		EMIOS_GFR[F12,F13]
907		EMIOS_GFR[F14,F15]
908		EMIOS_GFR[F16,F17]
909		EMIOS_GFR[F18,F19]
910		EMIOS_GFR[F20,F21]
911		EMIOS_GFR[F22,F23]
912		EMIOS_GFR[F24,F25]
913		EMIOS_GFR[F26,F27]
914		EMIOS_GFR[F28,F29]
915		EMIOS_GFR[F30,F31]

Table 26. Interrupt sources (continued)

Number	Description	Register bit
916	eMIOS_1	EMIOS_GFR[F0,F1]
917		EMIOS_GFR[F2,F3]
918		EMIOS_GFR[F4,F5]
919		EMIOS_GFR[F6,F7]
920		EMIOS_GFR[F8,F9]
921		EMIOS_GFR[F10,F11]
922		EMIOS_GFR[F12,F13]
923		EMIOS_GFR[F14,F15]
924		EMIOS_GFR[F16,F17]
925		EMIOS_GFR[F18,F19]
926		EMIOS_GFR[F20,F21]
927		EMIOS_GFR[F22,F23]
928		EMIOS_GFR[F24,F25]
929		EMIOS_GFR[F26,F27]
930		EMIOS_GFR[F28,F29]
931		EMIOS_GFR[F30,F31]
935	CCCU	cu_int

7.3.5.3 Interrupt latency

The Interrupt system is designed to provide a very short latency between the assertion of an interrupt request from a peripheral and the first instruction of the Interrupt Service Routine.

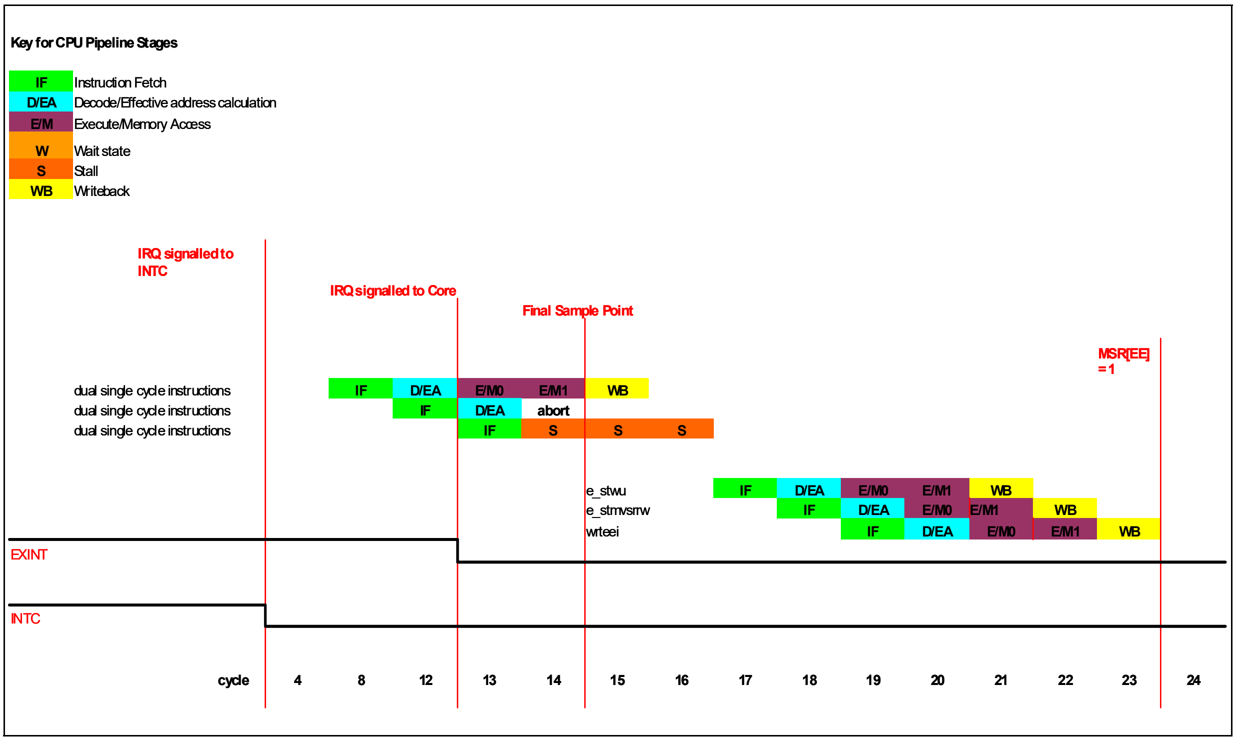
The latency time is difficult to define exactly because it depends on what interrupts the INTC is processing and what the CPU is doing.

In general, the latency time is divided into three subsections:

- INTC Latency
  - 3 clock cycles (INTC clock domain[b]) from the assertion of the interrupt request from the peripheral to assertion of the interrupt request to the CPU
- CPU Latency
  - 4 clock cycles (CPU clock domain[b]) from the assertion of the interrupt request from the INTC to the execution of the first instruction
- Software Latency
  - 7 clock cycles (CPU clock domain[b]) to execute the three instructions needed to save essential registers and enable interrupts

b. 120 MHz

Figure 11. Interrupt timing



7.3.6 DMA controller configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

Figure 12. DMA controllers configuration

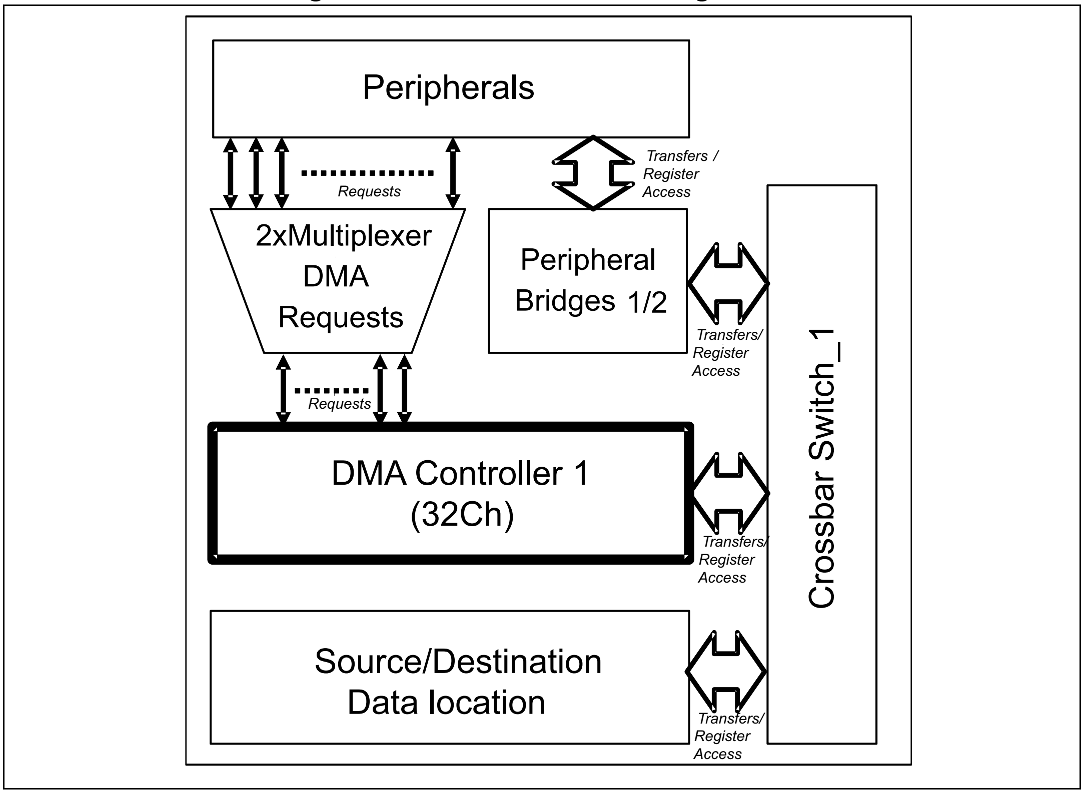


Table 27. Reference links to related information

Topic	Related module	Reference
Full description	DMA Controller 1	*Chapter 23: Enhanced Direct Memory Access (eDMA)*
Full description	DMACHMUX 0/1/2/3	*Chapter 24: DMA channel multiplexer (DMACHMUX)*
System memory map	—	*Chapter 5: Memory Map*
Register access	Peripheral bridge 1/2	*Chapter 19: Peripheral Bridge (PBRIDGE)*
Clocking	—	*Chapter 25: Clocking*
Transfers	Crossbar switch 1	*Chapter 17: Crossbar switch (XBAR)*

7.3.6.1 DMA implemented registers

Table 28 shows the registers implemented for each DMA instance on the SPC584Bx chip.

Table 28. DMA register comparison map

Register description	eDMA_1 Address offset (hex)
eDMA Control Register (eDMA_CR)	0x0000
eDMA Error Status (eDMA_ES)	0x0004

Table 28. DMA register comparison map (continued)

Register description				eDMA_1 Address offset (hex)
Reserved				0x0008
eDMA Enable Request Low (eDMA_ERQL, Channels 31-00)				0x000C
Reserved				0x0010
eDMA Enable Error Interrupt Low (eDMA_EEIL, Channels 31-00)				0x0014
eDMA Set Enable Request (eDMA_SERQ)	eDMA Clear Error (eDMA_CERR)	eDMA Set Enable Error Interrupt (eDMA_SEEI)	eDMA Clear Enable Error Interrupt (eDMA_CEEI)	0x0018
eDMA Clear Interrupt Request (eDMA_CINT)	eDMA Clear Error (eDMA_CERR)	eDMA Set Start Bit (eDMA_SSRT)	eDMA Clear Done Status Bit (eDMA_CDNE)	0x001C
Reserved				0x0020
eDMA Interrupt Request Low (eDMA_INTL, Channels 31-00)				0x0024
Reserved				0x0028
eDMA Error Low (eDMA_ERRL, Channels 31-00)				0x002C
Reserved				0x0030
eDMA Hardware Request Status Low (eDMA_HRSL, Channels 31-00)				0x0034
Reserved				0x0038–0x00FF
eDMA Channel 0 Priority (eDMA_DCHPRI0)	eDMA Channel 1 Priority (eDMA_DCHPRI1)	eDMA Channel 2 Priority (eDMA_DCHPRI2)	eDMA Channel 3 Priority (eDMA_DCHPRI3)	0x0100
eDMA Channel 4 Priority (eDMA_DCHPRI4)	eDMA Channel 5 Priority (eDMA_DCHPRI5)	eDMA Channel 6 Priority (eDMA_DCHPRI6)	eDMA Channel 7 Priority (eDMA_DCHPRI7)	0x0104
eDMA Channel 8 Priority (eDMA_DCHPRI8)	eDMA Channel 9 Priority (eDMA_DCHPRI9)	eDMA Channel 10 Priority (eDMA_DCHPRI10)	eDMA Channel 11 Priority (eDMA_DCHPRI11)	0x0108
eDMA Channel 12 Priority (eDMA_DCHPRI12)	eDMA Channel 13 Priority (eDMA_DCHPRI13)	eDMA Channel 14 Priority (eDMA_DCHPRI14)	eDMA Channel 15 Priority (eDMA_DCHPRI15)	0x010C
eDMA Channel 16 Priority (eDMA_DCHPRI16)	eDMA Channel 17 Priority (eDMA_DCHPRI17)	eDMA Channel 18 Priority (eDMA_DCHPRI18)	eDMA Channel 19 Priority (eDMA_DCHPRI9)	0x0110
eDMA Channel 20 Priority (eDMA_DCHPRI20)	eDMA Channel 21 Priority (eDMA_DCHPRI21)	eDMA Channel 22 Priority (eDMA_DCHPRI22)	eDMA Channel 23 Priority (eDMA_DCHPRI23)	0x0114
eDMA Channel 24 Priority (eDMA_DCHPRI24)	eDMA Channel 25 Priority (eDMA_DCHPRI25)	eDMA Channel 26 Priority (eDMA_DCHPRI26)	eDMA Channel 27 Priority (eDMA_DCHPRI27)	0x0118

Table 28. DMA register comparison map (continued)

Register description				eDMA_1 Address offset (hex)
eDMA Channel 28 Priority (eDMA_DCHPRI28)	eDMA Channel 29 Priority (eDMA_DCHPRI29)	eDMA Channel 30 Priority (eDMA_DCHPRI30)	eDMA Channel 31 Priority (eDMA_DCHPRI31)	0x011C
Reserved				0x0120-0x0FFF
TCD00-TCD15				0x1000-0x11FF
TCD16-TCD31				0x1200-0x13FF
Reserved				0x1400-0x18FF

7.3.7 DMACHMUX configuration

The peripheral DMA requests are serviced by a DMA controller. A static multiplexing scheme allows the support of more peripheral requests than available DMA channels.

7.3.7.1 DMACHMUX parameters

SPC584Bx has two DMA channel multiplexers.

Each DMACHMUX includes 16 channels.

Table 29 lists the number of DMA channels, always slots and DMA triggers.

Table 29. DMACHMUX controller generics

Signal name description	Description	DMACHMUX_0	DMACHMUX_1
NUMBER_OF_DMA_CHANNELS	Number of DMA channels	16	16
NUMBER_OF_ALWAYS_SLOTS	Module-internal slots which will always do a request	2	2
NUMBER_OF_DMA_TRIGGERS	Number of DMA channels which will be gated with a trigger pulse	3	3

7.3.7.2 DMA channel assignment

Table 30 lists which channels apply to each multiplexer.

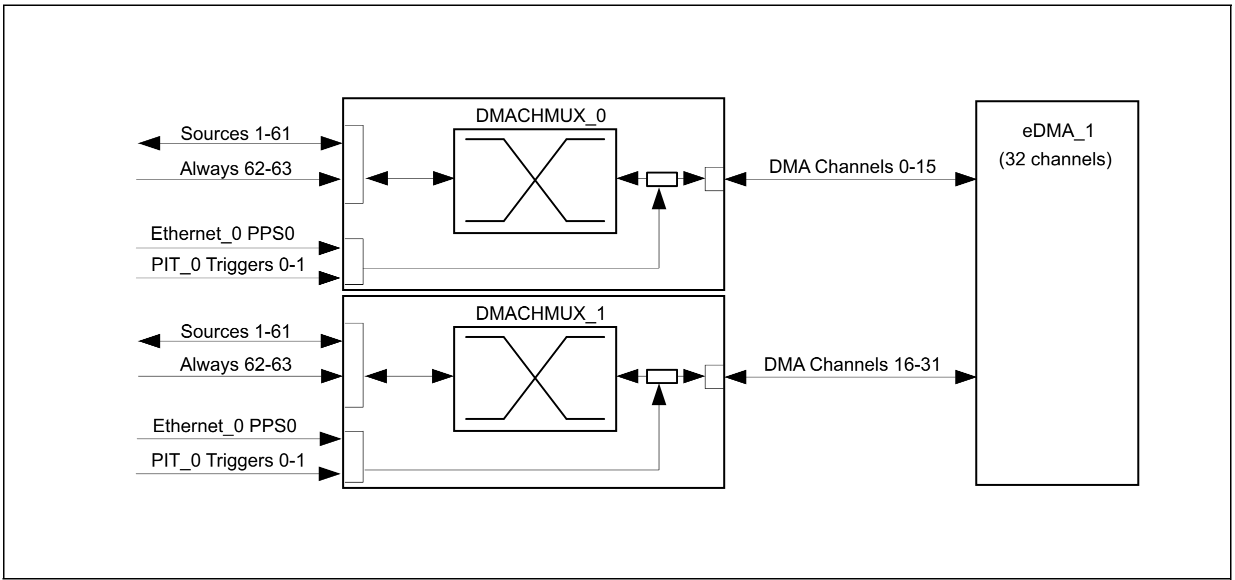
Table 30. SPC584Bx multiplexer and channel list

DMACHMUX instance	eDMA_1 channel
0	0-15
1	16-31

7.3.7.3 DMA channel multiplexing block diagram

The connections of the DMA channel multiplexers to the eDMA are shown in *Figure 13*. One DMACHMUX is instantiated for every sixteen DMA channels.

Figure 13. DMA channel multiplexing



7.3.7.4 DMA request mapping

The mapping of the peripheral DMA requests to the DMACHMUX input sources is shown in *Table 31*.

Table 31. DMAMUX Map

Channel	eDMA_1 (32 channels)	
	DMACHMUX_0 (16 DMA channels) source name	DMACHMUX_1 (16 DMA channels) source name
0	Not available	Not available
1	ADC_SAR_12bit SUPERVISOR EOC	ADC_SAR_12bit_1 EOC
2	ADC_SAR_12bit_0 EOC	ADC_SAR_10bit_STBY EOC
3	DSPI_0 RX	DSPI_4 RX
4	DSPI_0 TX	DSPI_4 TX
5	DSPI_0 CMD	DSPI_4 CMD
6	DSPI_1 RX	DSPI_5 RX
7	DSPI_1 TX	DSPI_5 TX
8	DSPI_1 CMD	DSPI_5 CMD
9	DSPI_2 RX	DSPI_6 RX
10	DSPI_2 TX	DSPI_6 TX

Table 31. DMAMUX Map (continued)

Channel	eDMA_1 (32 channels)	
	DMACHMUX_0 (16 DMA channels) source name	DMACHMUX_1 (16 DMA channels) source name
11	DSPI_2 CMD	DSPI_6CMD
12	DSPI 3_RX	LINFlexD 3_RX
13	DSPI 3_TX	LINFlexD 3_TX
14	DSPI 3_CMD	LINFlexD_4 RX
15	LINFlexD_0 RX	LINFlexD_4 TX
16	LINFlexD_0 TX	LINFlexD_5 RX
17	LINFlexD_1 RX	LINFlexD_5 TX
18	LINFlexD_1 TX	LINFlexD_6 RX
19	LINFlexD_2 RX	LINFlexD_6 TX
20	LINFlexD_2 TX	LINFlexD_8 RX
21	LINFlexD_7 RX	LINFlexD_8 TX
22	LINFlexD_7 TX	LINFlexD_9 RX
23	LINFlexD_10 RX	LINFlexD_9 TX
24	LINFlexD_10 TX	eMIOS_1 CH0
25	LINFlexD_15 RX	eMIOS_1 CH9
26	LINFlexD_15 TX	eMIOS_1 CH17
27	I2C_0 RX	eMIOS_1_CH18
28	I2C_0 TX	eMIOS_1 CH25
29	MCAN_1 (sys 0)	eMIOS_1 CH26
30	MCAN_2 (sys 0)	I2C_0 RX
31	SIUL2_REQ0	I2C_0 TX
32	SIUL2_REQ1	MCAN 1 (sys 0)
33	SIUL2_REQ2	MCAN_2 (sys 0)
34	SIUL2_REQ3	SIUL2_REQ0
35	eMIOS_0 CH0	SIUL2_REQ1
36	eMIOS_0 CH1	SIUL2_REQ2
37	eMIOS_0 CH9	SIUL2_REQ3
38	eMIOS_0 CH18	eMIOS_0 CH0
39	eMIOS_0 CH25	eMIOS_0 CH1
40	eMIOS_0 CH26	eMIOS_0 CH9
41	SIUL2_REQ4	LINFlexD_11RX

Table 31. DMAMUX Map (continued)

Channel	eDMA_1 (32 channels)	
	DMACHMUX_0 (16 DMA channels) source name	DMACHMUX_1 (16 DMA channels) source name
42	SIUL2_REQ5	LINFlexD_11TX
43	—	LINFlexD_12RX
44	—	LINFlexD_12TX
45	—	—
46	—	—
47	—	—
48	—	—
49	—	—
50	—	—
51	—	—
52	—	—
53	—	—
54	—	—
55	—	—
56	—	—
57	—	—
58	—	—
59	—	—
60	—	—
61	—	—
62	Always on	Always on
63	Always on	Always on
Trigger	PIT_0 CH0	PIT_0 CH0
	PIT_0 CH1	PIT_0 CH1
	Ethernet_0 PPS	Ethernet_0 PPS

Note: Just as multiple CHCFG registers within the same instance must not select the same source value, CHCFG registers among multiple instances must not select the same source value.

The mapping of the periodic triggers to the DMA channel sources is shown in *Table 32*.

Table 32. Periodic trigger to DMA channel mapping

Periodic trigger source	DMA channel
PIT_0 channel 0	0, 16
PIT_0 channel 1	1, 17
Ethernet PPS	2, 18

7.3.7.5 DMACHMUX Memory map

The 2 instances of DMACHMUX (0–1) are distributed on two PBRIDGE slots. For the DMACHMUX instances addresses, refer to *Chapter 5: Memory Map*.

The DMACHMUX address offsets are shown in *Table 33*.

Table 33. DMACHMUX channel configuration register address offsets

Channel Configuration register	DMACHMUX_0 address offset	DMACHMUX_1 address offset
CHCFG0	0x0000	0x0000
CHCFG1	0x0001	0x0001
CHCFG2	0x0002	0x0002
CHCFG3	0x0003	0x0003
CHCFG4	0x0004	0x0004
CHCFG5	0x0005	0x0005
CHCFG6	0x0006	0x0006
CHCFG7	0x0007	0x0007
CHCFG8	0x0008	0x0008
CHCFG9	0x0009	0x0009
CHCFG10	0x000A	0x000A
CHCFG11	0x000B	0x000B
CHCFG12	0x000C	0x000C
CHCFG13	0x000D	0x000D
CHCFG14	0x000E	0x000E
CHCFG15	0x000F	0x000F

Table 34. Reference links to related information

Topic	Related module	Reference
Full description	DMA Channel Multiplexer	*Chapter 24: DMA channel multiplexer (DMACHMUX)*
DMA controller description	DMA Controller	*Chapter 23: Enhanced Direct Memory Access (eDMA)*
System memory map	—	*Chapter 5: Memory Map*

7.3.8 Platform Configuration Module (PCM) configuration

The Platform Configuration Module (PCM) contains three registers. These PCM registers are used for controlling different aspects of the Platform functionality including the Frequency Gasket Configurations of the Intelligent Bus Bridging Gaskets.

The PCM module is mapped to PBRIDGE_2 on platform slot 10. Its base address is 0xF4028000.

7.3.8.1 Platform Configuration Module 0 register (PCM0)

Offset: 0x0000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	PRE_CONC1	BRE_CONC1	BWE_CONC1	0	0	0	PRE_DMA	BRE_DMA	BWE_DMA	0
W																
Reset	1	1	1	1	1	1	0	1	1	0	0	0	0	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Figure 14. Platform Configuration Module 0 register (PCM0)

Table 35. PCM0 field descriptions

Field	Description
14 BWE_DMA	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for DMA (DMAGasket) This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
13 BRE_DMA	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for DMA (DMAGasket) This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
12 PRE_DMA	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for DMA (DMAGasket) This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.

Table 35. PCM0 field descriptions (continued)

Field	Description
8 BWE_CONC1	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for Concentrator 1 (Concentr8tr1) This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
7 BRE_CONC1	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for Concentrator 1 (Concentr8tr1) This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
6 PRE_CONC1	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for Concentrator 1 (Concentr8tr1) This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.

7.3.8.2 Platform Configuration Module 1 register (PCM1)

Offset: 0x0004 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	PRE_HSM	BRE_HSM	BWE_HSM	0	0	EN_HSIZE_MUX_PBRIDGE	0	MSTR_STALL_NXMC_OVF_B	PRE_PBRIDGE2	BRE_PBRIDGE2	BWE_PBRIDGE2	PRE_PBRIDGE1	BRE_PBRIDGE1
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BWE_PBRIDGE1	PRE_ZXC_BD	BRE_ZXC_BD	BWE_ZXC_BD	IPS_CGM_INTC1	IPS_CGM_SMPU1	IPS_CGM_AXBS	IPS_CGM_PRAM3	IPS_CGM_PRAM2	IPS_CGM_PFL1	0	0	0	0	0	0
W																
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1

Figure 15. Platform Configuration Module 1 register (PCM1)

Table 36. PCM1 field descriptions

Field	Description
25 IPS_CGM_PFL1	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
24 IPS_CGM_PRAM2	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
23 IPS_CGM_PRAM3	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
22 IPS_CGM_AXBS	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
21 IPS_CGM_SMPU1	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
20 IPS_CGM_INTC1	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
19 BWE_ZXC_BD	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for ZXA TCM Backdoor access (Backdoor2Gasket) This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
18 BRE_ZXC_BD	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for ZXC TCM Backdoor access (Backdoor2Gasket) This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.

Table 36. PCM1 field descriptions (continued)

Field	Description
17 PRE_ZXC_BD	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for ZXC TCM Backdoor access (Backdoor2Gasket) This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
16 BWE_PBRIDGE1	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for PBRIDGE1 (PBRIDGE1 Gasket) This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
15 BRE_PBRIDGE1	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE1 (PBRIDGE1 Gasket) This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
14 PRE_PBRIDGE1	Pending read enable (PRE) used for controlling the urst parameter of frequency gasket used for PBRIDGE1 (PBRIDGE1 Gasket) This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
13 BWE_PBRIDGE2	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for PBRIDGE2 (PBRIDGE2 Gasket) This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
12 BRE_PBRIDGE2	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE2 (PBRIDGE2 Gasket) This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
11 PRE_PBRIDGE2	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE2 (PBRIDGE2 Gasket) This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
10 MSTR_STALL_ NXMC_OVF_B	Used for controlling the burst parameter of frequency gasket used for PBRIDGE2 (PBRIDGE2 Gasket) This bit controls the stalling of Masters when trace buffer in NXMC is full 0 Master is stalled when NXMC overflow occurs 1 Master is NOT stalled when NXMC overflow occurs

Table 36. PCM1 field descriptions (continued)

Field	Description
8 EN_HSIZE_MUX_ PBRIDGE	Force HSIZE=32 bit for Decorated Memory operations on PBRIDGE, as decorated operations compute write checkbits (ECC) on 32-bit word. DMC operations HSIZE control for PBRIDGE 1 and 2 0 During PBRIDGE write ECC check for decorated instructions wil be done with native HSIZE of the instruction (8b/16b/32b) 1 During PBRIDGE write ECC check for decorated instructions will be forced to use HSIZE of 32 bit
5 BWE_HSM	Burst write enable (BWE) used for controlling the burst parameter of frequency Gasket used for HSM (HSMGasket) This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
4 BRE_HSM	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for HSM (HSMGasket) This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
3 PRE_HSM	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for HSM (HSMGasket) This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.

7.3.8.3 Platform Configuration Module 2 register (PCM2)

Offset: 0x0008 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	EN_HSIZE_MUX_PRAM	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 16. Platform Configuration Module 2 register (PCM2)

Table 37. PCM2 field descriptions

Field	Description
9 EN_HSIZE_MUX_ PRAM	DMC operations HSIZE control for PRAM 2 and 3 Force HSIZE=32 bit for Decorated Memory operations on PRAM as decorated operations compute write checkbits (ECC) on 32-bit word. 0 Any decorated write to SRAM peripherals will be done with native HSIZE of the instruction (8b/16b/32b) 1 Any decorated write to SRAM peripherals will be done with HSIZE=32bit

7.3.9 Wakeup unit (WKPU) configuration

The Wakeup unit supports 24 external sources that can generate CPU exceptions or wakeup events and 2 external sources, driven by PA[4] and PD[15], that can cause non-maskable interrupt requests or wakeup events.

The effect of the signals coming from PA[4] and PD[15] depends on the execution mode.

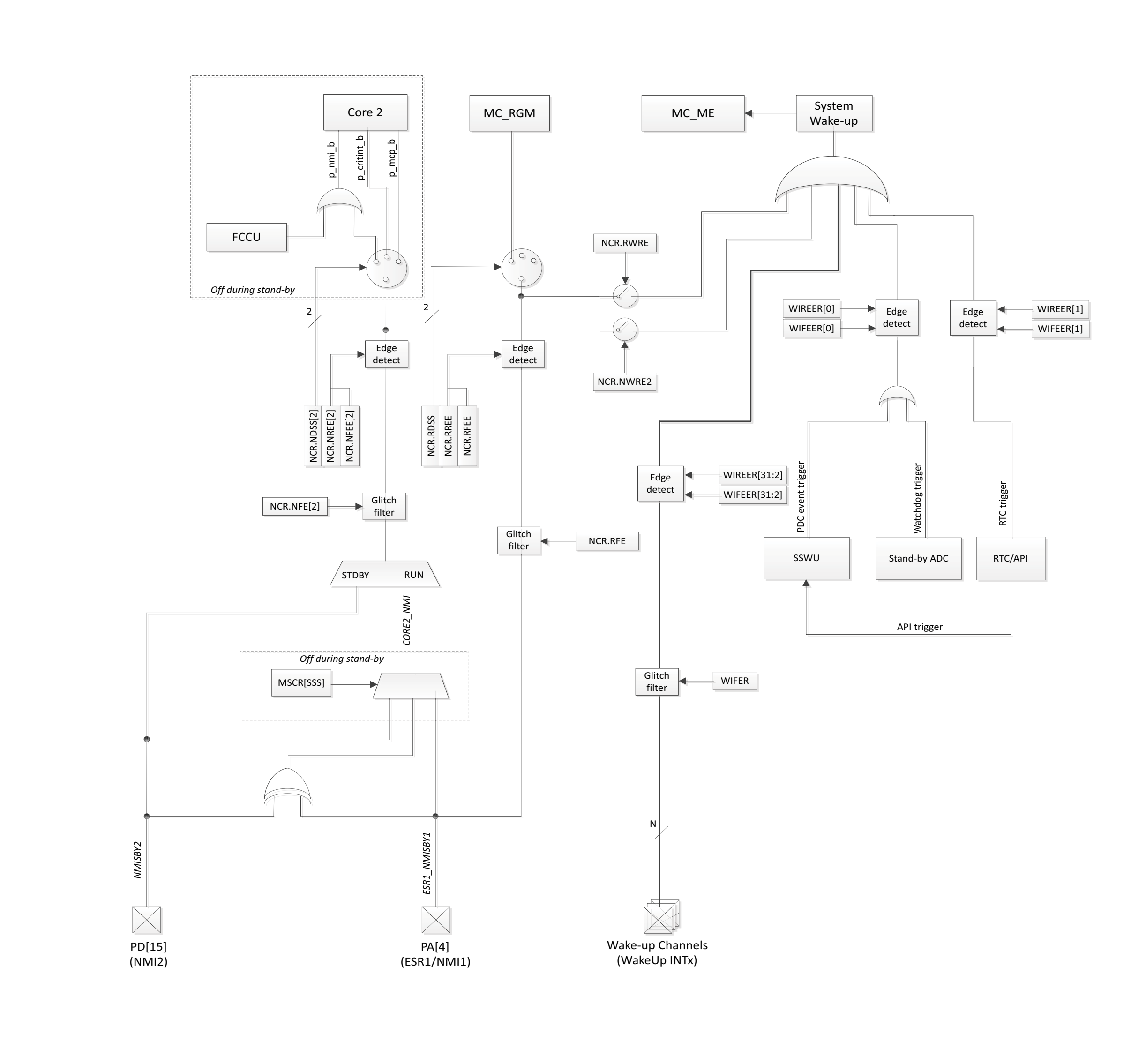
In any run mode, one signal (CORE2_NMI) is input to the core. This signal is the output of one multiplexer, which makes it possible to select (via SIUL2 register) any of the following sources:

- PA[4]
- PD[15]
- Logical XOR'ed of PA[4] and PD[15]

In stand-by mode, the signals driven by PD[15] (NMISBY2) can be used to wake-up the system.

PA[4] can be used to trigger a reset to the whole system.

Figure 17. NMI block diagram SPC584Bx



In *Figure 1622: Wakeup unit block diagram* of *Chapter 80: Wakeup unit (WKPU)*, it has to be noted that only one filter is implemented.

7.3.9.1 Features

The Wakeup unit supports these distinctive features:

- Non-maskable Interrupt support with:
  - 2 NMI source with bypassable glitch filter
  - Independent interrupt destination: non-maskable interrupt, critical interrupt, or machine check request
  - Edge detection
- External wakeup/interrupt support with:
  - 30 wakeup/interrupt sources
  - Analog glitch filter per each wakeup line
  - Independent interrupt mask
  - Edge detection
  - Configurable system wakeup triggering from all interrupt sources
  - Configurable pull/padkeeper functionality

Note: WIPUER register is used to disable Pad keeper functionality only on a set of interrupt pads (Group_1) and to enable internal pull only on a different set of interrupt pads (Group_2). Group_1 gathers : WKUP8 (PC1) - WKUP9 (PF3) - WKUP15 (PD5) - WKUP18 (PA2) - WKUP24 (PA11) - WKUP25 (PE10)) and Group_2 gathers all other WKUPx.

Note that the actual bit configuration for field WIPUER.IPUEx (x = 8/9/15/18/24/25) is:
1 Pad Keeper is disabled.
0 Pad Keeper is enabled.

- On-chip wakeup support
  - 2 wakeup sources
  - Wakeup status mapped to same register as external wakeup/interrupt status
  - Independent interrupt mask
  - Configurable system wakeup triggering from all interrupt sources

7.3.9.2 Wakeup vector mapping

Table 38. Wakeup vector mapping

Wakeup number	Port	Port Input Function	WKPU IRQ to INTC	IRQ#	WISR	Register Bit position(1)
WKUP0	API/SSWU	Refer to SPC584Bx pinout excel file	WakeUp_IRQ_0	Table 26: Interrupt sources	EISR0	31
WKUP1	RTC				EISR1	30
WKUP2	Refer to SPC584Bx pinout excel file				EISR2	29
—					Reserved	28
WKUP4					EISR4	27
WKUP5					EISR5	26
WKUP6					EISR6	25
—					Reserved	24
WKUP8			WakeUp_IRQ_1	Table 26: Interrupt sources	EISR8	23
WKUP9					EISR9	22
WKUP10					EISR10	21
WKUP11					EISR11	20
WKUP12					EISR12	19
WKUP13					EISR13	18
WKUP14					EISR14	17
WKUP15					EISR15	16
WKUP16			WakeUp_IRQ_2	Table 26: Interrupt sources	EISR16	15
WKUP17					EISR17	14
WKUP18					EISR18	13
WKUP19					EISR19	12
WKUP20					EISR20	11
WKUP21					EISR21	10
WKUP22					EISR22	9
WKUP23					EISR23	8
WKUP24			WakeUp_IRQ_3	Table 26: Interrupt sources	EISR24	7
WKUP25					EISR25	6
WKUP26					EISR26	5
WKUP27					EISR27	4
WKUP28					EISR28	3
WKUP29					EISR29	2
WKUP30		—			EISR30	1
WKUP31		—			EISR31	0

1. Applicable for registers WISR, IRER, WRER, WIFEER, WIFEEF, WIFER, WIPUER.

7.3.10 Crossbar integrity checker (XBIC)

The device implements the following XBIC instances:

- XBIC_Concentrator_1
- XBIC_1

Master and slave assignments are defined as follows:

- XBIC master definition follows the AHB master (see AHB Master ID assignments in *Section 7.3.2.2: AHB Master ID Assignments*).
- XBIC slave definition follows the XBAR slave numbering (see Crossbar_1 switch slave assignments in *Section 7.3.2.1.2: Crossbar_1 switch slave assignments*)

7.3.11 Body Cross Triggering Unit (BCTU) configuration

This section summarizes the Body Cross Triggering Unit (BCTU) configuration in the SPC584Bx device. For a comprehensive description of the BCTU, please refer to the BCTU's dedicated chapter.

7.3.11.1 Module overview

One single BCTU instance (BCTU0) is implemented. It is interconnected to:

- SAR_ADC_12bit_B
- eMIOS
- PIT

The Body Cross Triggering Unit is used to prioritize the trigger events from eMIOS/PIT.

The BCTU synchronizes the ADC conversion with PWM signal coming from eMIOS/PIT (refer to *Table 39*). When the event is programmed to be an ADC trigger, the 7-bit channel number is provided to ADC. This channel number is used to communicate which particular channel has to be converted (refer to *Table 40*).

The BCTU is also used to route the trigger events from eMIOS as a DMA request in case the event is not programmed to be a trigger to ADC.

The supported trigger sources are the following:

- 62 eMIOS channels
- 2 PIT channels

Figure 18. BCTU overview

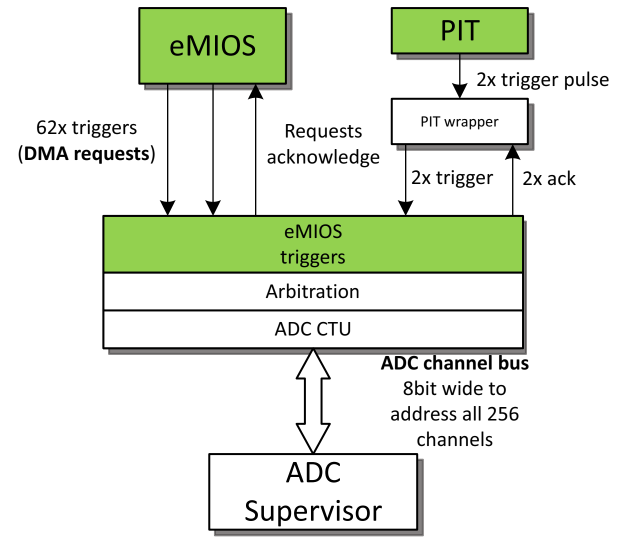


Table 39. BCTU input triggers allocation

Body CTU trigger input	Trigger source	Connected signal
0–22	eMIOS0	eMIOS_0 CH0–CH22
23	PIT_RTI_0	PIT_0 CH1
24–31	eMIOS_0	eMIOS_0 CH24–CH31
32–54	eMIOS_1	eMIOS_1 CH0–CH22
55	PIT_RTI_1	PIT_1 CH1
56–63	eMIOS_1	eMIOS_1 CH24–CH31

Table 40. Channel value ADC mapping

SAR_ADC_B	BCTU channel number
Internal channels	
0	0x00
1	0x01
...	...
63	0x3F
External channels	
128	0x40
129	0x41
...	...
191	0x7F

7.3.12 SSWU configuration

7.3.12.1 ADC channel connections

Table 41. Stand-by ADC internal channel connection (Params[6] = 0)

Params[6]	Params[5:0]	SSWU ADC channel
0	0	Reserved
	1	Reserved
	2	ADC SAR 10b Internal Channel 72
	3	ADC SAR 10b Internal Channel 73
	4	Reserved
	5	ADC SAR 10b Internal Channel 75
	6	ADC SAR 10b Internal Channel 76
	7	ADC SAR 10b Internal Channel 77
	8	Reserved
	9	ADC SAR 10b Internal Channel 79
	10	ADC SAR 10b Internal Channel 80
	11	ADC SAR 10b Internal Channel 81
	12	ADC SAR 10b Internal Channel 82
	13	Reserved
1	0	ADC SAR 10b External Channel 128
	1	ADC SAR 10b External Channel 129
	2	ADC SAR 10b External Channel 130
	3	ADC SAR 10b External Channel 131
	4	ADC SAR 10b External Channel 132
	5	ADC SAR 10b External Channel 133
	6	ADC SAR 10b External Channel 134
	7	ADC SAR 10b External Channel 135

7.4 Clocking

See *Chapter 25: Clocking* for a description of the architecture for the system level clocks including clock generation, the clock sources and the peripheral clocks. The chapter also provides configuration details for the Clock Monitor Unit (CMU).

7.5 Memories and memory interfaces

See the following chapters for detailed information on memories and memory interfaces.

Table 42. Reference links to related information

Topic	Related module	Reference
Embedded memories	—	Chapter 3: Embedded memories
System memory map	—	Chapter 5: Memory Map
Platform Flash	—	Chapter 34: Flash memory controller
Platform RAM controller	—	Chapter 33: Platform RAM controller AHB (PRAMC_AHB)
Embedded flash memory	—	Chapter 35: Embedded Flash Memory
Decorated Storage Memory Controller	—	Chapter 37: Decorated Storage Memory Controller (DSMC)

7.5.1 Flash memory controller (PFLASH) configuration

One flash controller and memory array is implemented. The flash controllers provide flash configuration and control functions and manage the interface between the flash memory array and the crossbar switch. The configuration and control functions are accessed via control registers, which are read/write accessible only in supervisor mode.

Three of the registers, PFCR1, PFCR2, and PFAPR, control the interaction of master modules with the flash array by enabling/disabling prefetch or by controlling access to the flash array on a per-master basis. Use of the fields in these registers requires knowledge of the number assigned to each master. For example, the PFAPR[M2AP] field controls flash array access by master 2, the PFAPR[M3AP] field controls flash array access by master 3, and so on. See the crossbar switch configuration section for details. See the PFCR3 register details in the flash memory controller chapter for more details on configuring the flash controller.

Table 43. Reference links to related information

Topic	Related module	Reference
Full description	Flash Memory Controller (PFLASH)	Chapter 34: Flash memory controller
Flash memory architecture overview	Flash memory, Flash controller	Section 3: Embedded memories
Flash memory	Flash memory	Chapter 35: Embedded Flash Memory
Flash port assignments	Crossbar switch	Section 7.3.2: Crossbar switch
System memory map	—	Chapter 5: Memory Map

The master assignments are shown in *Table 23: SMPU1 logical bus master assignments*.

7.5.2 Decorated Storage Memory Controller (DSMC)

The DSMC supports five store and three load operations, including:

- Bit field inserts
- Compare-and-store
- Bitwise AND, OR, and XOR operators
- Simple memory load
- Swap
- Load-and-set-1(bit)

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

The DSMC is instantiated within the core platform and physically resides between the CPU data AHB bus and the associated crossbar master port.

Due to this architecture choice, the DSMC can access the whole addressing space.

7.6 Analog modules

This section highlights the following analog modules that are implemented on SPC584Bx:

Table 44. Reference links to related information

Module or Subsystem	Device-specific information	Block details
ADC subsystem	*Section 38.1: ADC overview*	*Chapter 38: Analog-to-Digital Converters (ADC) Configuration*
Successive Approximation Register Analog-to-Digital Converters (SARADCs)	*Section 7.6.1: SARADC*	*Chapter 39: Successive Approximation Register Analog-to-Digital Converter (SARADC)*

The temperature sensor is used to measure the temperature of the SPC584Bx device. The temperature sensor has the following outputs:

- Two analog voltages
  - A voltage signal that is linearly increasing with internal junction temperature (PTAT that is proportional to absolute temperature)
  - A voltage signal that is linearly decreasing with internal junction temperature (CTAT that is complementary proportional to absolute temperature)
- Three digital outputs that signal under- or over-temperature operating conditions.

During production testing of the device, the output voltages of the temperature sensor are sampled by SARB of the onboard ADC module at a predefined high temperature and also at a predefined low temperature. These calibration parameters are stored during production test into UTEST Flash memory. See Section UTEST in Chapter Flash Memory.

By converting the two analog outputs of the temperature sensor through ADC, one can find out a constant reference code which does not change with temperature or the ADC reference values. It is also called 'Digital Bandgap Voltage'. See *Section 40.3.2: Equations* *for converting TSENS voltages into constant reference (Digital Bandgap Voltage)* in *Chapter 40: Temperature Sensor*.

For details, see *Chapter 40: Temperature Sensor*.

7.6.1 SARADC

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for an application. The SARADC provides accurate and fast conversion data for a wide range of applications. Each SARADC analog block has a corresponding digital interface.

SPC584Bx includes four (4) independent 12-bit SARADCs and one (1) independent 10-bit SARADC. The 12-bit SARADC_B has access to all external analog input pins, access to the external multiplexed inputs, and has special safety features.

For details, refer to the section Analog-to-Digital Converters (ADC) Configuration.

7.7 Timers

7.7.1 System Timer Module (STM) configuration

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the HPBM_CLK divided by an 8-bit prescale value (1 to 256).

SPC584Bx includes STM with four 32-bit compare channels.

Table 45. System Timer Module (STM) instances

Instance	Description	Number of channels	Number of registers (32-bit)
STM_2	Clocked by HPBM_CLK (Core_2)	4	14

7.7.2 Software Watchdog Timer (SWT) configuration

SPC584Bx includes two SWTs.

Table 46. Software Watchdog Timer (SWT) instances

Instance	Description
SWT2	Clocked by IRCOSC clock; using value of IAC8 register of Core 2 for instruction address compare feature
SWT3	Clocked by IRCOSC clock; not supporting the address compare feature

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Internal 16 MHz RC oscillator clock that is used for timer operation
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits

7.7.2.1 IAC (Instruction Address Compare)

The IAC feature makes it possible to serve a SWT when the CPU executes an instruction at a specific address location. The purpose is to guarantee that the application regularly executes an instruction within some fundamental section of the code. If this address location is not executed regularly, it implies that a critical error has occurred and the device should be eventually reset.

Not all the SWTs embed the IAC feature: *Table 46* shows for which watchdog the IAC feature is enabled.

If the IAC feature is enabled for a specific SWT, the address to be executed is set by the Instruction Address Compare 8 (IAC8) register, whereas the IAC feature can be activated by the register field SWT_CR[SMD] to one of the following settings:

- 2'b10 for Fixed Address Execution; the IAC8 register cannot be updated while the watchdog is enabled. If the SWT is in 'Fixed Address Execution' mode, an SWT output signal tells the core; this 'freezes' changes to the IAC8 register when the SWT is enabled
- 2'b11 for Incremental Address Execution; the IAC8 register can be updated

7.7.2.2 Reset assertion

The SWT can assert a reset when the watchdog timer expires. See *Section 8: Reset and Boot* chapter for details.

7.7.2.3 Default configuration

On SPC584Bx, the SWTs come out of reset with the default reset values summarized in *Table 47*. Minimum timeout value is 256 cycles for SWT0, SWT2, and SWT3. For details, please refer to *Section 52.3.5: Initialization of BAF DCF clients*

Table 47. Register reset values

SWT instance	Register	Reset value
SWT_2	SWT Control Register (SWT_CR)(1)	0xFF00_011B
SWT_2	SWT Time-out (SWT_TO)	0x0003_FDE0
SWT_3	SWT Control Register (SWT_CR)	0xFF00_010A
SWT_3	SWT Time-out (SWT_TO)	0x0005_FCD0

1. The SWT can be disabled by accepting a DCI-driven signal to clear the SWT_CR[WEN] bit. For SWT2, the default reset value for that bit is ‘1’, but the DCI can override that default value by setting it to ‘0’. If the signal is de-asserted, there is no change to the SWT_CR[WEN] value. (The DCI can only disable the SWT; it cannot enable it.) For details refer to *Section 58.1.3: Operating modes*.

7.7.3 PIT configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

Table 48. Reference links to related information

Topic	Related module	Reference
Full description	PIT	*Chapter 44: Periodic Interrupt Timer (PIT)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 25: Clocking*
Power management	—	*Chapter 10: Power management*
DMA channel assignments	DMACHMUX	*Chapter 24: DMA channel multiplexer (DMACHMUX)*
BCTU channel assignment	BCTU	*Chapter 41: Body Cross Triggering Unit (BCTU)*

7.7.3.1 PIT instantiation

Table 49. PIT information

PIT module	PIT channels	Description
PIT_0	8 + RTI	The PIT has eight standard channels. All PIT registers are cleared by any reset.
PIT_1	2	PIT1 has chaining mode to implement a 64-bit timer and the timer value is unchanged by a functional reset.

7.7.3.2 PIT/DMA periodic trigger assignments

The PIT generates periodic trigger events to the DMACHMUX. To check the mapping of the PIT channel number with the corresponding DMA channel number, refer to the *Table 32: Periodic trigger to DMA channel mapping*.

7.7.3.3 PIT Registers

The following tables show the implementation of PIT registers in this MCU.

Table 50. PIT_0 memory map

Offset address (hex)	Register name
0x0000	PIT Module Control Register (PIT0_MCR)(1)
0x00E0–0x00E4	Reserved
0x00F0–0x00FC	RTI Timer Channel Registers (PIT0_RTI_LDVAL, PIT0_RTI_CVAL, PIT0_RTI_TCTRL, PIT0_RTI_TFLG)
0x0100–0x017C	Timer Channel [0:7] Registers (PIT0_LDVALn, PIT0_CVALn, PIT0_TCTRLn, PIT0_TFLGn)
0x0180–0x3FFF	Reserved

1. The reset value of MCR is 0x06.

Table 51. PIT_1 memory map

Offset address (hex)	Register name
0x0000	PIT Module Control Register (PIT1_MCR)(1)
0x00E0	PIT Upper Lifetime Timer Register (PIT1_LTMR64H)
0x00E4	PIT Lower Lifetime Timer Register (PIT1_LTMR64L)
0x00F0–0x00F8	Reserved
0x0100–0x011C	Timer Channel [0:1] Registers (PIT1_LDVALn, PIT1_CVALn, PIT1_TCTRLn, PIT1_TFLGn)
0x0120–0x3FFF	Reserved

1. The reset value of MCR is 0x02.

7.7.3.4 PIT clocking

The PIT_0 standard channels are clocked with the peripheral clock (PER_CLK).

The PIT_0_RTI channel clock source is selectable between XOSC and RCOSC via the auxiliary clock selector 9.

The PIT_1 channels are clocked with the peripheral clock (PER_CLK).

Refer to *Figure 453: Block diagram of the PIT* for PIT clock sources description.

7.8 Communication interfaces

7.8.1 Ethernet configuration

7.8.1.1 Ethernet interface selection

On this chip, the Ethernet interface depends on the SIUL2_SCR0 settings (in the SIUL2), as shown in *Table 52*.

Table 52. Ethernet interface selection

SIUL2_SCR0[Ethernet_MODE]	Interface used by the Ethernet
0	RMII
1	MII

7.8.1.2 Ethernet timestamp snapshot trigger

The Ethernet interface includes an internal 64-bit time base counter to support IEEE 1588 Precision Time Protocol (PTP) timestamps.02

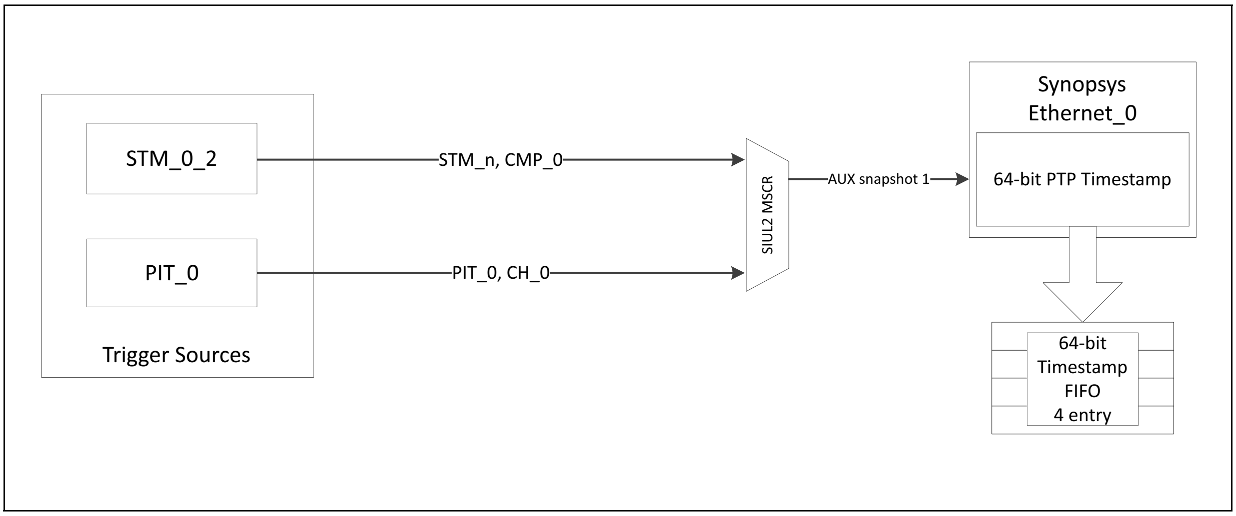
To synchronize other system events to the Ethernet PTP time base snapshot feature is available to capture the Ethernet time base on different system events into a four-entry-deep

FIFO. Details about the snapshot feature is available in *Section 47.3.9: IEEE 1588 Timestamps*.

The selection of the various trigger sources is done in the SIUL2 module by configuration of the corresponding MSCR registers. The details about the signal configuration are available in I/O Signal Table.

The system implementation of the snapshot feature is shown in *Figure 19*.

Figure 19. Ethernet Snapshot feature integration

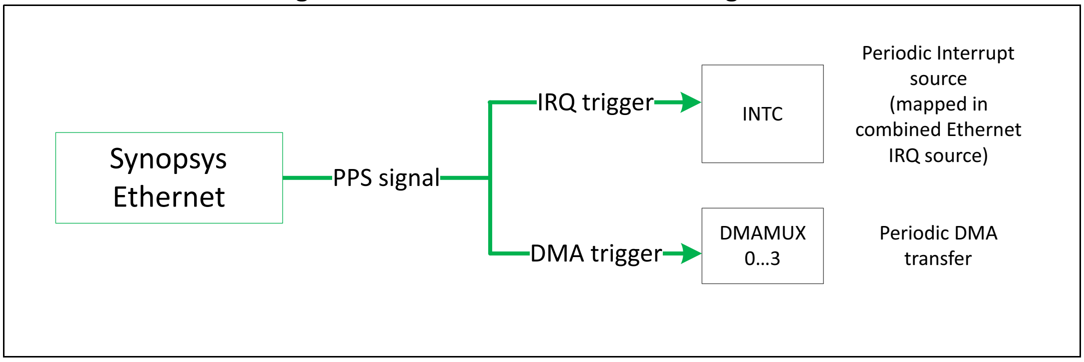


7.8.1.3 Ethernet Pulse-Per-Second output (PPS)

The Ethernet interface module includes a programmable time reference signal which can be used to trigger several system events. Details about the Pulse-Per-Second feature are available in *Section 47.3.9.4: Reference Timing Source*.

The system implementation of the PPS feature is shown in *Figure 20*.

Figure 20. Ethernet PPS feature integration



7.8.2 CAN subsystem configuration

Two CAN subsystems are implemented:

- CAN subsystem 0 (M_CAN type)
- CAN subsystem 1 (M_CAN type)

Please refer to *Chapter 46: CAN subsystem* for details.

7.8.2.1 CAN nodes

Table 53 lists the different CAN nodes for both CAN subsystems.

Note: All CAN instances are supporting ISO CAN FD mode.

Table 53. CAN bus controllers

Subsystem	Peripheral bridge instance	Shared message RAM size (words)	Type	Instance	ISO CAN FD support up to 64 data byte[(1)]	Debug on CAN support	Clock calibration[(2)]
CAN subsystem 0	PBRIDGE_2	3968	M_CAN	CAN_SUB_0_M_CAN_0	Yes	No	Yes
				CAN_SUB_0_M_CAN_1	Yes	Yes	Yes
				CAN_SUB_0_M_CAN_2	Yes	Yes	Yes
				CAN_SUB_0_M_CAN_3	Yes	No	Yes
CAN subsystem 1	PBRIDGE_1	3968	M_CAN	CAN_SUB_1_M_CAN_1	Yes	No	No
				CAN_SUB_1_M_CAN_2	Yes	No	No
				CAN_SUB_1_M_CAN_3	Yes	No	No
				CAN_SUB_1_M_CAN_4	Yes	No	No

1. M_CAN revision 3.2.0
2. Clock calibration based on M_CAN_1 messages for CAN subsystem 0.

7.8.3 DSPI configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module’s dedicated chapter.

Table 54. Reference links to related information

Topic	Related module	Reference
Full description	DSPI	*Chapter 49: Deserial Serial Peripheral Interface (DSPI)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 25: Clocking*
Signal Multiplexing	Port control	*Chapter 4: Signal Description*
System Module	System Integration Unit Lite	*Chapter 16: System Integration Unit Lite2 (SIUL2)*
DMA Channel Assignments	Direct Memory Access multiplexer	*Chapter 24: DMA channel multiplexer (DMACHMUX)*

7.8.3.1 DSPI instantiation

This device contains seven DSPI modules. *Table 55* shows the instantiation information for each module.

Table 55. DSPI instantiation

DSPI Module	CTARs	TX FIFO Depth	RX FIFO Depth	PCS Signals	Data Serialization Support
DSPI0	8	4	4	Link to External I/O Signal excel table	—
DSPI1	8	4	4		—
DSPI2	8	4	4		—
DSPI3	8	4	4		—
DSPI4	8	4	4		64
DSPI5	8	4	4		64
DSPI6	8	4	4		—

7.8.3.2 DSPI half duplex operation

Half duplex operation is not supported within the DSPI module, but it is configured by connecting the SIN and SOUT together within the SIUL2 and operating the combined pad in open drain mode. Software must ensure that the SOUT signal is passive (high) during SIN reception. This can be done by reconfiguring the SIUL2 to disconnect the SOUT or by writing a data frame that is all '1's.

7.8.3.3 Serialized timed I/O

The timer sources and injection pin inputs to be DSPI serialized downstream outputs are selected through the SIUL2 and MSCR registers. Inversion of these signals is also handled in the SIUL2. Refer to *Chapter 16: System Integration Unit Lite2 (SIUL2)*, for details.

7.8.3.4 DSPI HT external trigger

External hardware triggers are not supported.

7.8.3.5 DSPI ITSB mode

For the ITSB trigger, the internal DSPI trigger (TRGPRD in DSICR1) is used. Use of external trigger is not supported.

7.8.3.6 DSPI registers

The absolute addresses of DSPI modules registers in this MCU are provided in *Chapter 5: Memory Map*.

7.8.3.7 DSPI clocking

Refer to the SPC584Bx Datasheet for the DSPI Baud rates.

7.8.4 LinFlexD

7.8.4.1 LINFlexD configurations

SPC584Bx has 14 instances of LINFlexD. Table 56 lists the configuration for each instance.

Table 56. LINFlexD configurations

Description	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_15
Number of filters implemented	16	0												
Number of Tx DMA channels	1 (TX_CH_NUM = 0)(1)													
Number of Rx DMA channels	1 (RX_CH_NUM = 0)(2)													
LIN operation mode	Master/Slave (slave = 1)	Master (slave = 0)												
Value of auto-synchronization	1	0												

1. Number of TX DMA channels = 2**TX_CH_NUM where ** stands for exponentiation.
2. Number of RX DMA channels = 2**RX_CH_NUM where ** stands for exponentiation.

7.8.4.2 LINFlexD implemented registers

Table 57 shows the registers implemented for each LINFlexD instance on the SPC584Bx chip.

Table 57. LINFlexD register comparison map

Register description	Address offset (hex)													
	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_15
LIN Control register 1 (LINCR1)	00	00												
LIN Interrupt Enable register (LINIER)	04	04												
LIN Status register (LINSR)	08	08												
LIN Error Status register (LINESR)	0C	0C												
UART Mode Control register (UARTCR)	10	10												
UART Mode Status register (UARTSR)	14	14												
LIN Timeout Control Status register (LINTCSR)	18	18												
LIN Output Compare register (LINOCR)	1C	1C												
LIN Timeout Control register (LINTOCR)	20	20												
LIN Fractional Baud Rate register (LINFBRR)	24	24												
LIN Integer Baud Rate register (LINIBRR)	28	28												
LIN Checksum Field register (LINCFR)	2C	2C												
LIN Control register 2 (LINCR2)	30	30												
Buffer Identifier register (BIDR)	34	34												
Buffer Data Register Least Significant (BDRL)	38	38												
Buffer Data Register Most Significant (BDRM)	3C	3C												

Table 57. LINFlexD register comparison map (continued)

Register description	Address offset (hex)													
	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_15
Identifier Filter Enable register (IFER)	40	40												
Identifier Filter Match index (IFMI)	44	44												
Identifier Filter Mode register (IFMR)	48	48												
Identifier Filter Control registers (IFCR0)	4C	—												
Identifier Filter Control registers (IFCR1)	50	—												
Identifier Filter Control registers (IFCR2)	54	—												
Identifier Filter Control registers (IFCR3)	58	—												
Identifier Filter Control registers (IFCR4)	5C	—												
Identifier Filter Control registers (IFCR5)	60	—												
Identifier Filter Control registers (IFCR6)	64	—												
Identifier Filter Control registers (IFCR7)	68	—												
Identifier Filter Control registers (IFCR8)	6C	—												
Identifier Filter Control registers (IFCR9)	70	—												
Identifier Filter Control registers (IFCR10)	74	—												
Identifier Filter Control registers (IFCR11)	78	—												
Identifier Filter Control registers (IFCR12)	7C	—												
Identifier Filter Control registers (IFCR13)	80	—												

Table 57. LINFlexD register comparison map (continued)

Register description	Address offset (hex)													
	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_15
Identifier Filter Control registers (IFCR14)	84	—												
Identifier Filter Control registers (IFCR15)	88	—												
Global Control register (GCR)	8C	8C												
UART Preset Timeout register (UARTPTO)	90	90												
UART Current Timeout register (UARTCTO)	94	94												
DMA Tx Enable register (DMATXE)	98	98												
DMA Rx Enable register (DMARXE)	9C	9C												

7.9 Reset and Boot modules

7.9.1 BAF configuration

7.9.1.1 Search boot header and boot options

Refer to *Section 8.3.12.2.3: SW boot record search (BAF search)* in the *Chapter 8: Reset and Boot*.

7.9.1.2 LINFlexD and MCAN pins configuration

Table 58 shows the device pin configurations for LINFlexD_0/2 and CAN_SUB_0_M_CAN_1/2 modules.

Table 58. LinFlexD and MCAN pin configuration

Pads	Initial Serial Boot Mode		Serial Boot Mode after a valid CAN message received	Serial Boot Mode after a valid LINFlexD message received
	Function	Pad Configuration	Function	Function
PA[10]	MCAN_2 Tx	(1) Output:— Push/Pull medium drive no pullup/pulldown	MCAN_2 Tx	LINFlexD_2 Tx
PA[11]	LINFlexD_2 Rx and MCAN_2 Rx	Input:— high-Z hysteresis CMOS level	MCAN_2 Rx	LINFlexD_2 Rx
PB[10]	MCAN_1 Tx	(1) Output:— Push/Pull medium drive no pullup/pulldown	MCAN_1 Tx	LINFlexD_0 Tx
PB[9]	LINFlexD_0 Rx and MCAN_1 Rx	Input:— high-Z hysteresis CMOS level	MCAN_1 Rx	LINFlexD_0 Rx

1. Pad is set to output configuration as described in the table. Output buffer is enabled for the whole duration of the BAF serial boot.

7.9.2 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration in the SPC584Bx device. For a comprehensive description of the SSCM, please refer to the SSCM's dedicated chapter.

7.9.2.1 SSCM instantiation

There is one SSCM instance in the SPC584Bx device. The SSCM is part of the reset and boot sub-system.

7.9.2.2 Device-specific features

The device specific reset values for the SSCM_MEMCONFIG register are shown in *Table 59*.

Table 59. SSCM_MEMCONFIG reset values

Field	Field definition	Value
JPIN	JTAG Part ID	10'h110(1)
MREV	Minor Revision	4'h0(2)

1. This value will change for each major revision of the device.
2. This value will change for each minor revision of the device.

Table 60 shows the address locations that the SSCM uses to search for startup information.

Table 60. SSCM startup information addresses

Description	Value
The location at which the SSCM will look for the UTEST DCF records (UTEST_OFFSET)	0x0040_0000
Location of the Boot Assist code in Flash (BAF)	0x0040_4000
Specifies the locations which are searched for a valid HSM boot header (starting from the lowest address)	0x60_C000 0x61_0000

7.9.2.3 Memory map and register description

Table 61. Reference text to related information

Topic	Related module	Reference
Reset	Reset Generation Module (MC_RGM)	*Chapter 51: Reset Generation Module (MC_RGM)*
Reset/Boot	Reset and Boot	*Chapter 8: Reset and Boot*
Boot	Boot Assist from Flash (BAF)	*Chapter 52: Boot Assist Flash (BAF)*
Register Protection	Register Protection (REG_PROT)	*Chapter 77: Register Protection (REG_PROT)*
Register Protection	Register Protection Configuration	Section SSCM registers protection in *Chapter 7: Device configuration*
Serial Boot	LINFlexD	*Chapter 50: LINFlexD*
Security	PASS	*Chapter 78: Password and Device Security Module (PASS)*

7.10 Safety modules

7.10.1 CRC

Table 62 lists the number of contexts available for the current CRC computation of multiple data streams of CRC.

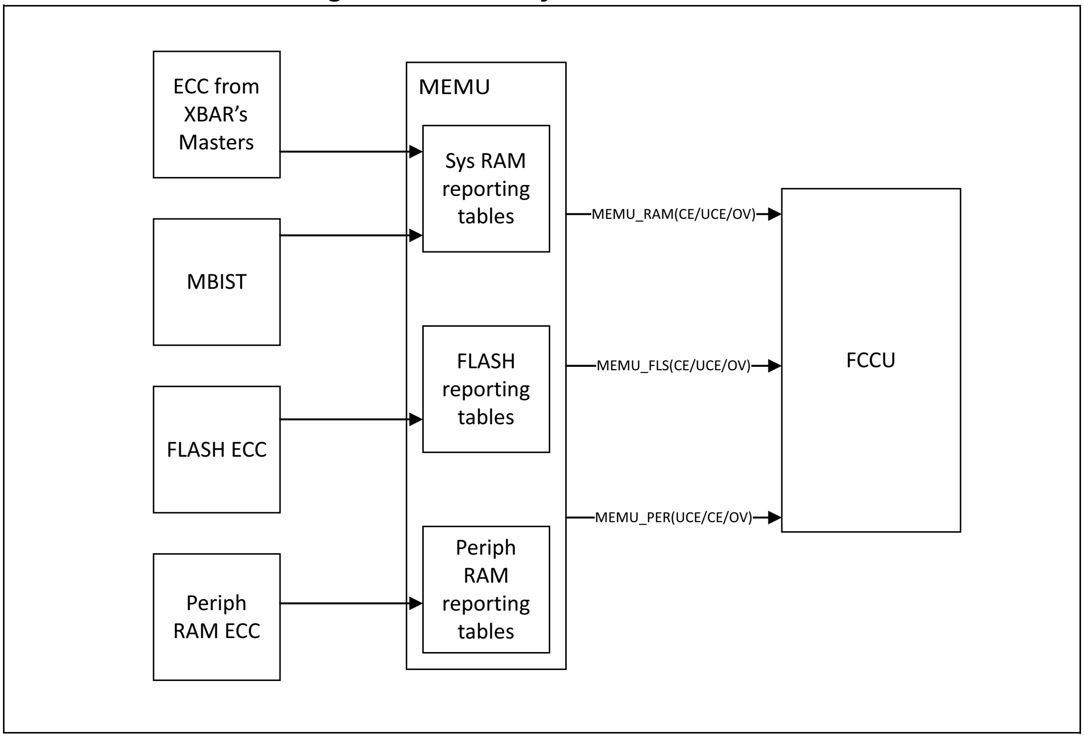
Table 62. CRC contexts

Signal name	Description	CRC_0	CRC_1
CRC_CNTX_NUM	Number of contexts available for the current CRC computation of multiple data streams.	4	4

7.10.2 MEMU configuration

7.10.2.1 MEMU system connections

Figure 21. MEMU system connections



7.10.2.2 MEMU error sources

Table 63 shows the MEMU error sources for each instance configured on this chip.

Table 63. MEMU error sources

Instance	Value
Number of system RAM unique error sources	73
Number of peripheral RAM unique error sources	16
Number of flash memory unique error sources	2

7.10.2.3 Reporting tables implementation

There are 3 categories of error reporting for the MEMU:

- Flash error reporting
- System RAM error reporting
- Peripheral RAM error reporting

Each category of error reporting has two tables associated with it, these are:

- Correctable Error Reporting Table
- Uncorrectable Error Reporting Table

Each entry in a reporting table corresponds to a unique error event. The number of entries supported by each table depends upon the error source.

The table below describes the number of entries in each reporting table for the different error sources.

Table 64. Number of entries in each reporting table

Error Source	Number of entries in correctable error reporting table	Number of entries in uncorrectable error reporting table
Flash	20	1
System RAM	10	1
Peripheral RAM	2	1

7.10.2.4 Concurrent overflow register (OFLW) implementation

Errors are reported to the MEMU from one of the three categories, Flash, System RAMs or Peripheral RAMs. In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLW register details the corresponding error source to the respective bit-field in the Concurrent Overflow registers.

Note: z4c stands for Core_2 in the following tables.

Table 65. Corresponding error sources for system RAM OFLW0 register

Source number	OFLW0 bit-field position (bit)(1)	Error sources
6	25	z4c Data AHB
7	24	z4c Instruction AHB
8	23	z4c Data TCM (DMEM)
9	22	—
10	21	z4c Data Cache
11	20	z4c Instruction Cache
12	19	—
13	18	—
14	17	—
15	16	—
16	15	—
17	14	—
18	13	—
19	12	SRAM Memory 2
20	11	-—
21	10	—
22	9	Decorated z4c
23	8	—

Table 65. Corresponding error sources for system RAM OFLW0 register (continued)

Source number	OFLW0 bit-field position (bit)(1)	Error sources
24	7	Concentrator 0 (DMA_1)
25	6	Concentrator 3 (HSM)
26	5	Concentrator 1 (SIPI_1 port)
27	4	
28	3	Concentrator_1 (Ethernet_0)
29	2	
30	1	
31	0	

1. Bit 0 in the table bit-field corresponds to bit 31 in target register

Table 66. Corresponding error sources for system RAM OFLW1 register

Source number	OFLW1 bit-field position (bit)(1)	Error sources
32	31	—
33	30	—
34	29	—
35	28	—
36	27	—
37	26	—
38	25	—
39	24	MBIST z4c Data Cache Instance 1
40	23	MBIST z4c Data Cache Instance 2
41	22	MBIST z4c Data Cache Instance 3
42	21	MBIST z4c Data Cache Instance 4
43	20	—
44	19	—
45	18	MBIST Ethernet EVEN
46	17	MBIST z4c Data TCM Instance 1
47	16	MBIST z4c Data TCM Instance 2
48	15	MBIST z4c Instruction Cache Instance 1
49	14	MBIST z4c Instruction Cache Instance 2
50	13	MBIST z4c Instruction Cache Instance 3
51	12	MBIST z4c Instruction Cache Instance 4
52	11	MBIST z4c Instruction Cache Tag

Table 66. Corresponding error sources for system RAM OFLW1 register (continued)

Source number	OFLW1 bit-field position (bit)(1)	Error sources
53	10	MBIST HSM RAM 0
54	9	MBIST HSM RAM 1 Instance 1
55	8	MBIST HSM RAM 1 Instance 2
56	7	MBIST HSM Instruction Cache Instance 1
57	6	MBIST HSM Instruction Cache Instance 2
58	5	MBIST HSM Instruction Cache Tag Instance 1
59	4	MBIST HSM Instruction Cache Tag Instance 2
60	3	MBIST HSM Crypto Channel Controller (C3)
61	2	—
62	1	—
63	0	—

1. Bit 0 in the table bit-field corresponds to bit 31 in target register

Table 67. Corresponding error sources for system RAM OFLW2 register

Source number	OFLW2 bit-field position (bit)(1)	Error sources
64	31	MBIST Controller Area Network 0
65	30	—
66	29	—
67	28	—
68	27	—
69	26	—
70	25	—
71	24	—
72	23	—
73	22	—
74	21	—
75	20	—
76	19	—
77	18	—
78	17	—
79	16	—
80	15	—
81	14	—

Table 67. Corresponding error sources for system RAM OFLW2 register (continued)

Source number	OFLW2 bit-field position (bit)(1)	Error sources
82	13	—
83	12	—
84	11	MBIST Ethernet RX 0 ODD
85	10	MBIST Ethernet TX 0 ODD
86	9	—
87	8	—
88	7	MBIST FEC TX 0 EVEN
89	6	—
90	5	—
91	4	—
92	3	—
93	2	MBIST MP Flash Krypton RAM
94	1	—
95	0	MBIST DMA 1

1. Bit 0 in the table bit-field corresponds to bit 31 in target register

Table 68. Corresponding error sources for system RAM OFLW3 register

Source number	OFLW3 bit-field position (bit)(1)	Error sources
96	31	—
97	30	—
98	29	MBIST Controller Area Network 1
99	28	—
100	27	—
101	26	—
102	25	—
103	24	—
104	23	—
105	22	—
106	21	—
107	20	—
108	19	—
109	18	MBIST SRAM Instance 2_0_0
110	17	MBIST SRAM Instance 2_0_1

Table 68. Corresponding error sources for system RAM OFLW3 register (continued)

Source number	OFLW3 bit-field position (bit)(1)	Error sources
111	16	MBIST SRAM Instance 2_1
112	15	MBIST SRAM Instance 2_2
113	14	MBIST SRAM Instance 2_3
114	13	—
115	12	—
116	11	—
117	10	—
118	9	MBIST ZXC Data Cache Tag
119	8	—
120	7	—
121	6	—
122	5	—
123	4	—
124	3	—
125	2	—
126	1	—
127	0	—

1. Bit 0 in the table bit-field corresponds to bit 31 in target register

Table 69. Corresponding error sources for peripheral RAM OFLW0 register

Source number	OFLW0 bit-field position (bit)(1)	Error sources
0	31	—
1	30	—
2	29	Ethernet RX_EVEN
3	28	CAN_RAM_1
4	27	—
5	26	AIPS BRIDGE 1
6	25	Ethernet TX EVEN
7	24	Ethernet RX ODD
8	23	HSM Platform RAM
9	22	HSM Instruction Cache RAM
10	21	—
11	20	—

Table 69. Corresponding error sources for peripheral RAM OFLW0 register (continued)

Source number	OFLW0 bit-field position (bit)(1)	Error sources
12	19	CAN RAM 0
13	18	—
14	17	—
15	16	—
16	15	—
17	14	—
18	13	—
19	12	—
20	11	—
21	10	—
22	9	—
23	8	—
24	7	—
25	6	DMA 1 TCD
26	5	—
27	4	—
28	3	AIPS BRIDGE 2
29	2	Ethernet TX ODD
30	1	—
31	0	—

1. Bit 0 in the table bit-field corresponds to bit 31 in target register

Table 70. Corresponding error sources for flash memory OFLW0 register

Source number	OFLW0 bit-field position (bit)(1)	Error sources
0	31	Reserved
1	30	Reserved
2	29	Flash 1 port 0
3	28	Flash 1 port 1
4	27	Reserved
5	26	Reserved
6	25	Reserved
7	24	Reserved
8	23	Reserved

Table 70. Corresponding error sources for flash memory OFLW0 register (continued)

Source number	OFLW0 bit-field position (bit)(1)	Error sources
9	22	Reserved
10	21	Reserved
11	20	Reserved
12	19	Reserved
13	18	Reserved
14	17	Reserved
15	16	Reserved
16	15	Reserved
17	14	Reserved
18	13	Reserved
19	12	Reserved
20	11	Reserved
21	10	Reserved
22	9	Reserved
23	8	Reserved
24	7	Reserved
25	6	Reserved
26	5	Reserved
27	4	Reserved
28	3	Reserved
29	2	Reserved
30	1	Reserved
31	0	Reserved

1. Bit 0 in the table bit-field corresponds to bit 31 in target register

7.10.2.5 MEMU registers

Table 71 shows the address offsets and implementation of the MEMU registers in this chip.

Table 71. MEMU registers

Offset	Register
0x0000	Control register (CTRL)
0x0004	Error flag register (ERR_FLAG)
0x000C	Debug register (DEBUG)
0x0020	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS0)

Table 71. MEMU registers (continued)

Offset	Register
0x0024	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR0)
0x0028	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS1)
0x002C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR1)
0x0030	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS2)
0x0034	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR2)
0x0038	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS3)
0x003C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR3)
0x0040	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS4)
0x0044	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR4)
0x0048	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS5)
0x004C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR5)
0x0050	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS6)
0x0054	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR6)
0x0058	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS7)
0x005C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR7)
0x0060	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS8)
0x0064	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR8)
0x0068	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS9)
0x006C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR9)
0x0070	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)
0x0074	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)
0x0078	System RAM concurrent overflow register (SYS_RAM_OFLW0)
0x007C	System RAM concurrent overflow register (SYS_RAM_OFLW1)
0x0080	System RAM concurrent overflow register (SYS_RAM_OFLW2)
0x0084	System RAM concurrent overflow register (SYS_RAM_OFLW3)
0x0620	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS0)

Table 71. MEMU registers (continued)

Offset	Register
0x0624	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR0)
0x0628	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS1)
0x062C	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR1)
0x0630	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)
0x0634	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)
0x0638	Peripheral RAM concurrent overflow register (PERIPH_RAM_OFLW0)
0x0C20	Flash memory correctable error reporting table status register (FLASH_CERR_STS0)
0x0C24	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR0)
0x0C28	Flash memory correctable error reporting table status register (FLASH_CERR_STS1)
0x0C2C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR1)
0x0C30	Flash memory correctable error reporting table status register (FLASH_CERR_STS2)
0x0C34	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR2)
0x0C38	Flash memory correctable error reporting table status register (FLASH_CERR_STS3)
0x0C3C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR3)
0x0C40	Flash memory correctable error reporting table status register (FLASH_CERR_STS4)
0x0C44	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR4)
0x0C48	Flash memory correctable error reporting table status register (FLASH_CERR_STS5)
0x0C4C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR5)
0x0C50	Flash memory correctable error reporting table status register (FLASH_CERR_STS6)
0x0C54	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR6)
0x0C58	Flash memory correctable error reporting table status register (FLASH_CERR_STS7)
0x0C5C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR7)
0x0C60	Flash memory correctable error reporting table status register (FLASH_CERR_STS8)
0x0C64	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR8)
0x0C68	Flash memory correctable error reporting table status register (FLASH_CERR_STS9)

Table 71. MEMU registers (continued)

Offset	Register
0x0C6C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR9)
0x0C70	Flash memory correctable error reporting table status register (FLASH_CERR_STS10)
0x0C74	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR10)
0x0C78	Flash memory correctable error reporting table status register (FLASH_CERR_STS11)
0x0C7C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR11)
0x0C80	Flash memory correctable error reporting table status register (FLASH_CERR_STS12)
0x0C84	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR12)
0x0C88	Flash memory correctable error reporting table status register (FLASH_CERR_STS13)
0x0C8C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR13)
0x0C90	Flash memory correctable error reporting table status register (FLASH_CERR_STS14)
0x0C94	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR14)
0x0C98	Flash memory correctable error reporting table status register (FLASH_CERR_STS15)
0x0C9C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR15)
0x0CA0	Flash memory correctable error reporting table status register (FLASH_CERR_STS16)
0x0CA4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR16)
0x0CA8	Flash memory correctable error reporting table status register (FLASH_CERR_STS17)
0x0CAC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR17)
0x0CB0	Flash memory correctable error reporting table status register (FLASH_CERR_STS18)
0x0CB4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR18)
0x0CB8	Flash memory correctable error reporting table status register (FLASH_CERR_STS19)
0x0CBC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR19)
0x0CC0	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)
0x0CC4	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)
0x0CC8	Flash memory concurrent overflow register (FLASH_OFLW0)

7.10.2.6 Indirect Memory Access (IMA) configuration

The IMA refers to the alteration of memory data values during system development and test activities. This section provides details about what memory locations can be accessed, either via processor core access or via the IMA module, and how they are configured. For details on IMA see *Chapter 74: Indirect Memory Access (IMA)*.

Note: See the SPC584Bx Microcontroller Security Reference Manual for the HSM IMA details.

7.10.2.7 SRAM Arrays accessible by processor cores

All the SRAM arrays in *Table 72* can be accessed by the core (Core_2).

Table 72. SRAM with processor Core access

Block	Function	SPC584Bx		
		Words	Bits	ECC Bits
Core_2	DMEM	8192	40	—
		8192	40	—
	Icache	256	72	—
		256	72	—
		256	72	—
		256	72	—
	Dcache	128	80	—
		128	80	—
		128	80	—
		128	80	—
	ITAG	128	65	—
	DTAG	64	71	—
System RAM 2	System RAM	1024	72	—
		3072	72	—
		4096	72	—
		4096	72	—
		4096	72	—

7.10.2.8 Memory accessible via the IMA module

The SRAM arrays in *Table 73* are accessible via the IMA Module.

Table 73. SRAM arrays accessible via IMA

IMA ARRAY SELECT	Block	Function	SPC584Bx		
			Words	Bits	ECC Bits
0	None	IMA Disabled	—	—	—
1	Reserved	Reserved	—	—	—
2	DMA	DMA_1	128	72	71–64
3	Reserved	Reserved	—	—	—
4			—	—	—
5			—	—	—
6			—	—	—
7	Reserved	Reserved	—	—	—
8			—	—	—
9	CAN Subsystem 0	CAN Subsystem 0	3968	39	38–32
10	Reserved	Reserved	—	—	—
11			—	—	—
12			—	—	—
13			—	—	—
14			—	—	—
15			—	—	—
16			—	—	—
17			—	—	—
18			—	—	—
19			—	—	—
20			—	—	—
21			—	—	—
22			—	—	—
23			—	—	—
24			—	—	—
25			—	—	—
26			—	—	—
27			—	—	—
28	Ethernet	Ethernet ODD RX	512	76	75–64
29		Ethernet ODD TX	256	76	75–64
30		Ethernet EVEN RX	512	76	75–64
31		Ethernet EVEN TX	256	76	75–64
32	CAN Subsystem 1	CAN Subsystem 1	3968	39	38–32

7.10.3 FCCU configuration

7.10.3.1 Available FCCU output signals

Table 74 shows the available FCCU output signals implemented on this chip.

Table 74. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0], FI[0], ERROR0	F0
EOUT[1], FI[1], ERROR1	F1

7.10.3.2 FCCU register reset values

Table 75 shows the reset values of selected FCCU registers.

Table 75. Reset values of selected FCCU registers

Register	Reset value
FCCU_RF_CFG0	0xFFFF_FFFF
FCCU_RF_CFG1	0xFFFF_FFFF
FCCU_RF_CFG2	0xFFFF_FFFF
FCCU_RF_CFG3	0xFFFF_FFFF
FCCU_RFS_CFG0	0x0000_2000
FCCU_RFS_CFG1	0x0000_0020
FCCU_RFS_CFG2	0x0000_0000
FCCU_RFS_CFG3	0x0000_0000
FCCU_RFS_CFG4	0x0000_0000
FCCU_RFS_CFG5	0x0000_0000
FCCU_RFS_CFG6	0x0000_0000
FCCU_RFS_CFG7	0x0000_0000
FCCU_RF_S0	0x0000_0000
FCCU_RF_S1	0x0000_0000
FCCU_RF_S2	0x0000_0000
FCCU_RF_S3	0x0000_0000
FCCU_RF_E0	0x0004_0040
FCCU_RF_E1	0x0000_0000
FCCU_RF_E2	0x0000_0000
FCCU_RF_E3	0x0000_0000
FCCU_RF_TOE0	0x0000_0000
FCCU_RF_TOE1	0x0000_0000
FCCU_RF_TOE2	0x0000_0000

Table 75. Reset values of selected FCCU registers (continued)

Register	Reset value
FCCU_RF_TOE3	0x0000_0000
FCCU_EOUT_SIG_EN0	0x0000_0000
FCCU_EOUT_SIG_EN1	0x0000_0000
FCCU_EOUT_SIG_EN2	0x0000_0000
FCCU_EOUT_SIG_EN3	0x0000_0000
FCCU_CTRL	0x0000_00C0
FCCU_CTRLK	0x0000_0000
FCCU_CFG	0x0000_0000
FCCU_RFK	0x0000_0000
FCCU_RF_T0	0x0000_FFFF
FCCU_CFG_T0	0x0000_0006
FCCU_EINOUT	0x0000_0030
FCCU_STAT	0x0000_0010
FCCU_NAFS	0x0000_0000
FCCU_AFFS	0x0000_0000
FCCU_NFFS	0x0000_0000
FCCU_FAFS	0x0000_0000
FCCU_SCFS	0x0000_0000
FCCU_RFF	0x0000_0000
FCCU_IRQ_STAT	0x0000_0000
FCCU_IRQ_EN	0x0000_0000
FCCU_XTMR	0x0000_0000
FCCU_MCS	0x0000_8083
FCCU_TRANS_LOCK	0x0000_0000
FCCU_PERMNT_LOCK	0x0000_0000
FCCU_DELTA_T	0x0000_0000
FCCU_IRQ_ALARM_EN0	0x0000_0000
FCCU_IRQ_ALARM_EN1	0x0000_0000
FCCU_IRQ_ALARM_EN2	0x0000_0000
FCCU_IRQ_ALARM_EN3	0x0000_0000
FCCU_NMI_EN0	0x0000_0000
FCCU_NMI_EN1	0x0000_0000
FCCU_NMI_EN2	0x0000_0000
FCCU_NMI_EN3	0x0000_0000

7.10.3.3 FOSU parameters

The FOSU_COUNT, or FOSU timeout response time, is 1000 IRCOSC clock cycles.

7.10.4 Register protection (REG_PROT) configuration

Some modules on this device include a built-in register protection mechanism that can be used to override the normal write permissions associated with individual registers in the module. This mechanism is software-controlled and can be dynamically reconfigured at runtime or made effective until device reset.For a detailed explanation, see the *Chapter 77: Register Protection (REG_PROT)*.

Table 76. Reference links to related information

Topic	Related module	Reference
System memory map	—	*Chapter 5: Memory Map*
System Integration Unit Lite2 registers	SIUL2	*Chapter 16: System Integration Unit Lite2 (SIUL2)*
Clock monitor Unit registers	CMU_1 CMU_0	*Chapter 27: Clock Monitor Unit (CMU)*
PMC Digital Interface registers	PMC_DIG	*Chapter 54: Power management controller digital interface (PMC_Dig)*
PLL Digital Interface registers	PLL_DIG	*Chapter 26: Dual PLL digital interface (PLLDIG)*
Oscillator Digital Interface registers	OSC40M_DIG	*Chapter 29: OSC digital interface (XOSC)*
IRCOSC digital interface registers	RC16M_DIG	*Chapter 30: IRCOSC digital interface*
Memory Error Management Unit registers	MEMU	*Chapter 73: Memory Error Management Unit (MEMU)*
Mode Entry registers	MC_ME	*Chapter 56: Mode Entry Module (MC_ME)*
Clock Generation Module registers	MC_CGM	*Chapter 28: Clock Generation Module (MC_CGM)*
Power Control Unit registers	MC_PCU	*Chapter 55: Power Control Unit (MC_PCU)*
Reset Generation Module registers	MC_RGM	*Chapter 51: Reset Generation Module (MC_RGM)*
System Status and Configuration Module registers	SSCM	*Chapter 53: System Status and Configuration Module (SSCM)*
Indirect Memory Access registers	IMA	*Chapter 74: Indirect Memory Access (IMA)*
JTAG Master registers	JTAGM	*Chapter 63: JTAG Master (JTAGM)*

7.10.4.1 SIUL2 protected registers

Table 77 lists the SIUL2 registers that can be protected.

Table 77. Protected SIUL2 registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
DISR0	32	0x0010	32
DIRER0	32	0x0018	32

Table 77. Protected SIUL2 registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
DIRSR0	32	0x0020	32
IREER0	32	0x0028	32
IFEER0	32	0x0030	32
IFER0	32	0x0038	32
IFMCR0–31	32	—(1)	32 (×32)
IFCPR	32	0x00C3	32
SCR0	32	0x0100	32
MSCR0–50	32	—(1)	32 (×51)
MSCR52–83	32	—(1)	32 (×31)
MSCR86–148	32	—(1)	32 (×62)
MSCR160–161	32	—(1)	32 (×2)
MSCR174	32	—(1)	32
MSCR184–192	32	—(1)	32 (×8)
MSCR194	32	—(1)	32
MSCR196	32	—(1)	32
MSCR198	32	—(1)	32
MSCR206–207	32	—(1)	32 (×2)
MSCR552–575	32	—(1)	32 (×24)
MSCR600–609	32	—(1)	32 (×10)
MSCR624–625	32	—(1)	32 (×2)
MSCR638–639	32	—(1)	32 (×2)
MSCR656–751	32	—(1)	32 (×96)
MSCR757–760	32	—(1)	32 (×4)
MSCR764–767	32	—(1)	32 (×4)
MSCR772–774	32	—(1)	32 (×3)
MSCR808–810	32	—(1)	32 (×3)
MSCR812	32	0x0EF0	32
MSCR848–860	32	—(1)	32 (×12)
MSCR863–865	32	—(1)	32 (×2)
MSCR880–900	32	—(1)	32 (×20)
MSCR922–932	32	—(1)	32 (×11)
MSCR934	32	—(1)	32
MSCR936–967	32	—(1)	32 (×32)
MSCR986–1017	32	—(1)	32 (×32)

Table 77. Protected SIUL2 registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
GPDO0–21	8	—(1)	8 (×22)
GPDO24–100	8	—(1)	8 (×77)
GPDO107–130	8	—(1)	8 (×23)
GPDO136–148	8	—(1)	8 (×12)
GPDO160–161	8	—(1)	8 (×2)
GPDO164–165	8	—(1)	8 (×2)
GPDO174	8	—(1)	8
GPDO184	8	—(1)	8
GPDO192–194	8	—(1)	8 (×3)
GPDO198	8	—(1)	8
GPDO206–207	8	—(1)	8 (×2)
PGPDO0–11	16	—(1)	16 (×12)
PGPDO13	16	—(1)	16

1. See the module memory map in the SIUL2 chapter for address offset.

7.10.4.2 Core Clock Monitor Unit (CMU_1) protected registers

Table 78 lists the fast crossbar CMU registers that can be protected.

Table 78. Protected CMU_1 (CORE_XBAR) registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CSR	32	0x0003	8
HFREFR	32	0x000A	16
LFREFR	32	0x000E	16
ISR	32	0x0013	8

7.10.4.3 I/O Processor Clock Monitor Unit (CMU_0) protected registers

Table 79 lists the PLL0 CMU registers that can be protected.

Table 79. Protected CMU_0 (PLL0_XOSC_IRCOSC) registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CSR	32	0x0003	8
HFREFR	32	0x000A	16
LFREFR	32	0x000E	16
ISR	32	0x0013	8

7.10.4.4 PMC Digital interface (PMC_DIG) protected registers

Table 80 lists the PMC_DIG registers that can be protected.

Table 80. Protected PMC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
EPR_LV0	32	0x0000	8
REE_LV0	32	0x0004	8
RES_LV0	32	0x0008	8
IE_LV0	32	0x000C	8
FEE_LV0	32	0x0010	8
EPR_LV1	32	0x0021	8
REE_LV1	32	0x0025	8
RES_LV1	32	0x0029	8
IE_LV1	32	0x002D	8
FEE_LV1	32	0x0031	8
EPR_HV0	32	0x0040	8
REE_HV0	32	0x0044	8
RES_HV0	32	0x0048	8
IE_HV0	32	0x004C	8
FEE_HV0	32	0x0050	8
EPR_HV1	32	0x0060	32
REE_HV1	32	0x0064	32
RES_HV1	32	0x0068	32
IE_HV1	32	0x006C	32
FEE_HV1	32	0x0070	32
IE_G	32	0x0088	8
VSIO	32	0x0107	8
PNREG_S	32	0x0243	8
MISC_CTRL_REG	32	0x0248	8
SSWU_CTRL_REG	32	0x024E	16
EPR_TD	32	0x0303	8
REE_TD	32	0x0307	8
RES_TD	32	0x030B	8
CTL_TD	32	0x030C	32
FEE_TD	32	0x031B	8
DREG_QVF_STATUS	32	0x036E	16
DREG_VREF_STATUS	32	0x0370	32

Table 80. Protected PMC_DIG registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
DREG_DELAY_STATUS	32	0x0376	16
DREG_SLOPE0	32	0x037A	16
DREG_SLOPE1	32	0x037E	16
DREG_SLOPE2	32	0x0382	16
DREG_SLOPE3	32	0x0386	16
DREG_SLOPE4	32	0x038A	16
DREG_SLOPE5	32	0x038E	16
DREG_SLOPE6	32	0x0392	16
DREG_SLOPE7	32	0x0396	16
DREG_SLOPE8	32	0x039A	16
DREG_SLOPE9	32	0x039E	16
DREG_SLOPE10	32	0x03A2	16
BIST_CTRL	32	0x03D4	32

7.10.4.5 PLL Digital interface (PLL_DIG) protected registers

Table 81 lists the PLL_DIG registers that can be protected.

Table 81. Protected PLL_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PLL0_CR	32	0x0000	32
PLL0_SR	32	0x0004	32
PLL0_DIVR	32	0x0008	32
PLL1_CR	32	0x0020	32
PLL1_SR	32	0x0024	32
PLL1_DIVR	32	0x0028	32
PLL1_FMR	32	0x002C	32
PLL1_FDR	32	0x0030	32

7.10.4.6 Oscillator Digital interface (OSC40M_DIG) protected registers

Table 82 lists the XOSC registers that can be protected.

Table 82. Protected OSC40M_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTL	32	0x0000	32

7.10.4.7 IRCOSC Digital interface (RC16M_DIG) protected registers

Table 83 lists the IRCOSC registers that can be protected.

Table 83. Protected RC16M_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTL	32	0x0001	8

7.10.4.8 Memory Error Management Unit (MEMU) protected registers

Table 84 lists the MEMU registers that can be protected.

Table 84. Protected MEMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTRL	8	0x0002	8
DEBUG	32	0x000C	32

7.10.4.9 Mode Entry (MC_ME) protected registers

Table 85 lists the ME registers that can be protected.

Table 85. Protected MC_ME registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MCTL	32	0x0004	32
ME	32	0x000A	16
IS	32	0x000F	8
IM	32	0x0013	8
IMTS	32	0x0017	8
SAFE_MC	32	0x0028	32
DRUN_MC	32	0x002C	32
RUN0–3_MC	32	0x0030–0x003C	32 (x4)
HALT0_MC	32	0x0040	32
STOP0_MC	32	0x0048	32
STANDBY0_MC	32	0x0054	32
RUN_PC0–7	32	0x0083–0x009F	8 (x8)
LP_PC0–7	32	0x00A2–0x00BE	8 (x8)
PCTL15	8	0x00CF	8
PCTL30	8	0x00DE	8
PCTL36	8	0x00E4	8

Table 85. Protected MC_ME registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PCTL38	8	0x00E6	8
PCTL62	8	0x00FE	8
PCTL67–69	8	0x0103–0x0105	8 (x3)
PCTL72	8	0x0108	8
PCTL74	8	0x010A	8
PCTL86–92	8	0x0115–0x011C	8 (x7)
PCTL96–99	8	0x0120–0x0123	8 (x4)
PCTL101	8	0x0125	8
PCTL107	8	0x012B	8
PCTL112	8	0x0130	8
PCTL114	8	0x0132	8
PCTL127	8	0x013F	8
PCTL129–130	8	0x0141–0x0142	8 (x2)
PCTL141	8	0x014D	8
PCTL143	8	0x014F	8
PCTL158	8	0x015E	8
PCTL164	8	0x0164	8
PCTL166	8	0x0166	8
PCTL195–198	8	0x0183–0x0186	8 (x4)
PCTL202	8	0x018A	8
PCTL213			8
PCTL215–220	8	0x0194–0x019C	8 (x6)
PCTL225–227	8	0x01A0–0x01A3	8 (x3)
PCTL234	8	0x01AA	8
PCTL240–243	8	0x01B0–0x01B3	8 (x4)
PCTL255	8	0x01BF	8
CCTL0	16	0x01C4	16
CCTL4	16	0x01CC	16
CADDR0	32	0x01E0	32
CADDR4	32	0x01F0	32

7.10.4.10 Clock Generation Module (MC_CGM) protected registers

Table 86 lists the CGM registers that can be protected.

Table 86. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SC_DIV_RC	32	0x01D3	8
DIV_UPD_TYPE	32	0x01D4	32
DIV_UPD_TRIG	32	0x01D8	32
DIV_UPD_STAT	32	0x01DC	32
SC_DC0–4	32	0x01E8–0x01F8	16 (x5)
AC0_SC	32	0x0200	8
AC0_DC0	32	0x0208	16
AC0_DC1	32	0x020C	16
AC1_SC	32	0x0220	8
AC1_DC0	32	0x0228	16
AC3_SC	32	0x0260	8
AC4_SC	32	0x0280	8
AC6_SC	32	0x02C0	8
AC6_DC0	32	0x02C8	16
AC8_SC	32	0x0300	8
AC8_DC0	32	0x0308	16
AC9_SC	32	0x0320	8
AC9_DC0	32	0x0328	16
AC11_SC	32	0x0360	8
AC11_DC0	32	0x0368	16
AC12_SC	32	0x0380	8
AC12_DC0	32	0x0388	32
AC12_DC1	32	0x038C	16
AC12_DC2	32	0x0390	16
AC12_DC3	32	0x0394	16

7.10.4.11 Power Control Unit (MC_PCU) protected registers

Table 87 lists the PCU registers that can be protected.

Table 87. Protected MC_PCU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PCONF0	32	0x0002	16
PCONF1	32	0x0006	16

Table 87. Protected MC_PCU registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PCONF2	32	0x000A	16
PCONF3	32	0x000E	16

7.10.4.12 Reset Generation Module (MC_RGM) protected registers

Table 88 lists the RGM registers that can be protected.

Table 88. Protected MC_RGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
DES	32	0x0000	32
DERD	32	0x0010	32
DEAR	32	0x0020	32
DBRE	32	0x0030	32
FES	32	0x0300	32
FERD	32	0x0310	32
FEAR	32	0x0320	32
FBRE	32	0x0330	32
FESS	32	0x0340	32
FRET	32	0x0604	8
DRET	32	0x0608	8
PRST0–7	32	0x0610–0x062C	32 (x8)

7.10.4.13 System Status and Control Module (SSCM) protected registers

Table 89 lists the SSCM registers that can be protected.

Table 89. Protected SSCM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
STATUS	16	0x0000	8
ERROR	16	0x0006	16

7.10.4.14 Indirect Memory Access (IMA) protected registers

Table 90 lists the IMA registers that can be protected.

Table 90. Protected IMA registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ENABLE	32	0x0004	32

7.10.4.15 JTAG Master (JTAGM) protected registers

Table 91 lists the JTAGM registers that can be protected.

Table 91. Protected JTAGM registers

Register	Register Size (bits)	Offset from Module Base Address	Protected Size (bits)
MCR	32	0x00	32

7.10.5 STCU2 configuration

The number of memory cuts (NMCUT) available in this device is 35.

Table 92. MBIST partitions

$MBIST_i$	Location	Partition	Addressable	Start address in MEMU	End address in MEMU
0	ROM	BAR	Yes	NA	NA
1		MPFLASH	No	NA	NA
2	FLASH	MPFLASH_1	No	0x4F1F_0000	0x4F1F_3FFF
3	ETHERNET	FEC_rx_0_even	No	0x4F0A_4000	0x4F0A_4FFF
4	Core_2	D-MEM (1)	Yes	0x5280_0000	0x5280_FFFF
		D-MEM (0)	Yes		
6		I_Cache (3)	No	0x4F08_0000	0x4F08_FFFF
7		I_Cache (2)	No		
8		I_Cache (1)	No		
9		I_Cache (0)	No		
10		ITAG	No	0x4F09_0000	0x4F09_FFFF
11	DMA	DMA	Yes	0xF40A_5000	0xF40A_7FFF
12	Platform	System RAM 2	Yes	0x400A_8000	0x400A_9FFF
13			Yes	0x400A_A000	0x400A_FFFF
14			Yes	0x400C_0000	0x400C_7FFF
15			Yes	0x400B_8000	0x400B_FFFF
16			Yes	0x400B_0000	0x400B_7FFF
17	CAN(1)	CAN_1	Yes	0xFBED_4000	0xFBED_7FFF
18		CAN_0	Yes	0xF7ED_4000	0xF7ED_7FFF

Table 92. MBIST partitions (continued)

$MBIST_i$	Location	Partition	Addressable	Start address in MEMU	End address in MEMU
19	ETHERNET	FEC_tx_0_even	No	0x4F0A_0000	0x4F0A_07FF
20	HSM(2)	pram_0	Yes	0xA000_8000	0xA000_9FFF
21		pram_1_1	Yes	0xA000_4000	0xA000_7FFF
22		pram_1_0	Yes	0xA000_0000	0xA000_3FFF
23		I_Cache (1)	No	0x4F0B_0000	0x4F0B_0FFF
24		I_Cache (0)	No		
25		I_TAG (1)	No	0x4F0C_0100	0x4F0C_01FF
26		I_TAG (0)	No	0x4F0C_0000	0x4F0C_00FF
27		C3	Yes	0xA3F8_0000	0xA3F8_3FFF
28	ETHERNET_0	FEC_rx_0_odd	No	0x4F0A_7000	0x4F0A_7FFF
29		FEC_tx_0_odd	No	0x4F0A_1000	0x4F0A_17FF
30	Core_2	D_Cache (3)	No	0x4F05_0000	0x4F05_FFFF
31		D_Cache (2)			
32		D_Cache (1)			
33		D_Cache (0)			
34		DTAG	No	0x4F07_0000	0x4F07_FFFF

1. The clock domain for CAN memories is supplied by PFBRIDGE_CLK. In case of offline MBIST which runs at speed (120 MHz), then CAN memories works at 30 MHz (CGM_SC_DC4 = 4 by default, thus PFBRIDGE_CLK = SYS_CLK/4). To test the CAN memories at the maximum frequency (120 MHz when PFBRIDGE_CLK = SYS_CLK/1), the online MBIST has to use programming CGM_SC_DC4 = 1.
2. HSM and FLASH online MBIST are not accessible when lifecycle is customer delivered/OEM/IN FIELD (SSCM_LCSTAT[LC]=CUST_DELIV/PROD_OEM/IN_FIELD).

Table 93 lists the MBIST registers in STCU2 with their available fields.

Table 93. MBIST registers in STCU2 with available fields

MBIST register	Fields available
STCU_MBS1	MBS31 – MBS0
STCU_MBS2	MBS34 – MBS32
STCU_MBE1	MBE31 – MBE0
STCU_MBE2	MBE34 – MBE32
STCU_MBS1SW	MBSSW31 – MBSSW0
STCU_MBS2SW	MBSSW34 – MBSSW32
STCU_MBE1SW	MBESW31 – MBESW0
STCU_MBE2SW	MBESW34 – MBESW32
STCU_MBUFM1	MBUFM31 – MBUFM0
STCU_MBUFM2	MBUFM34 – MBUFM32

Note: Refer to the Self-Test Control Unit (STCU2) chapter for a description of the registers programming and to the DCF client table for the description of the available DCF. The STCU setup has to be defined based on the application-specific safety, boot time and consumption requirements. By default, STCU2 is not bypassed. This means that if offline self tests are not programmed, the hard-coded WDG of the STCU2 will be anyway activated. As a consequence, the startup time of the device will be increased accordingly to the WDG timeout value.

7.11 Security modules

7.11.1 Password and Device Security Module (PASS) configuration

The PASS module is used to implement password-based on several resources like the Flash R/W access, or the debugger JTAG port. Up to four levels of password protection can be implemented for each resource.

Note: The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

See *Chapter 78: Password and Device Security Module (PASS)* for further details.

7.11.1.1 PASS_LOCK*n*_PG*n* register bit mapping

Each password group defined in the DCF records has a set of four LOCK*n* registers association with it: PASS_LOCK*n*_PG*n.* The bits of these registers are associated with specific flash blocks. The mapping is shown in the following figures.

Note: The mapping of various LOCK register bits to flash blocks in the SPC584Bx microcontroller is very similar to the LOCKx registers implemented in the FLASH, but not identical. The PASS module has additional mapping in the LOCK3 register and defines read locking regions. See the PASS chapter for details.

Bit	Register field	Flash block name	Read lock region
0	TSLOCK	16 KB UTEST NVM Block Space	Read Lock Region 0
1	Reserved	—	—
2	LOWLOCK[13:0]	—	—
3		—	—
4		—	—
5		—	—
6		—	—
7		—	—
8		128 KB HSM Code block0	Read Lock Region 3
9		32 KB Code Flash block2[(1)]	Read Lock Region 1
10		32 KB Code Flash block1[(1)]	
11		32 KB Code Flash block0[(1)]	
12		16 KB HSM Code block3	Read Lock Region 3
13		16 KB Code Flash block2[(1)]	Read Lock Region 1
14		16 KB Code Flash block1[(1)]	
15		16 KB BAF block0	—
16	MIDLOCK[15:0]	—	—
17		—	—
18		—	—
19		—	—
20		—	—
21		—	—
22		—	—
23		—	—
24		—	—
25		—	—
26		—	—
27		—	—
28		—	—
29		—	—
30		16 KB HSM EEPROM block1	Read Lock Region 4
31		16 KB HSM EEPROM block0	

1. See *Chapter 35: Embedded Flash Memory* for block addresses,

Figure 22. PASS_LOCK0_PG*n* Registers

Read lock region	Flash block name	Register field	Bit
—	—	Reserved	0
—	—		1
—	—		2
—	—		3
—	—		4
—	—		5
—	—		6
—	—		7
—	—		8
—	—		9
—	—		10
—	—		11
—	—		12
—	—		13
—	—		14
—	—		15
—	—	HIGHLOCK[15:0]	16
—	—		17
—	—		18
—	—		19
—	—		20
—	—		21
—	—		22
—	—		23
—	—		24
—	—		25
—	—		26
—	—		27
Read Lock Region 2	16 KB FMC EEPROM Block3		28
	16 KB FMC EEPROM Block2		29
	16 KB FMC EEPROM Block1		30
	16 KB FMC EEPROM Block0		31

Figure 23. PASS_LOCK1_PG*n* registers

Bit	Register field	Flash block name	Read lock region
0	A256KLOCK[31:0]	—	—
1		—	—
2		—	—
3		—	—
4		—	—
5		—	—
6		—	—
7		—	—
8		—	—
9		—	—
10		—	—
11		—	—
12		—	—
13		—	—
14		—	—
15		—	—
16		—	—
17		—	—
18		—	—
19		—	—
20		—	—
21		—	—
22		—	—
23		—	—
24		256 KB Code Flash block6(1)	Read Lock Region 1
25		256 KB Code Flash block5(1)	
26		256 KB Code Flash block4(1)	
27		256 KB Code Flash block3(1)	
28		256 KB Code Flash block2(1)	
29		256 KB Code Flash block1(1)	
30		256 KB Code Flash block0(1)	
31		128 KB Code Flash block0(1)	

1. See *Chapter 35: Embedded Flash Memory* for block addresses,

Figure 24. PASS_LOCK2_PG*n* registers

Bit	Register field	Flash block name
0	PGL	Password Group Lock
1	DBL	Debug Interface Lock
2	Reserved	—
3	Reserved	—
4	Reserved	—
5	Reserved	—
6	Reserved	—
7	Reserved	—
8	Reserved	—
9	Reserved	—
10	Reserved	—
11	RL4	HSM EEPROM Data Flash Read Lock
12	RL3	HSM Code Flash Read Lock
13	RL2	EEPROM Data Flash Read Lock
14	RL1	Code Flash Read Lock
15	RL0	UTEST Read Lock
16	A256KLOCK[47:32]	—
17		—
18		—
19		—
20		—
21		—
22		—
23		—
24		—
25		—
26		—
27		—
28		—
29		—
30		—
31		—

Figure 25. PASS_LOCK3_PG*n* registers

7.11.1.2 PASS Module DCF Records

The records required to configure the initial values for the PASS module password group registers are shown in *Table 94*. These values provide the following capabilities:

- Set flash memory to censored/uncensored state (censorship also depends on lifecycle)
- Override the basic flash write protection for flash memory blocks
- Set levels of password-based write protection (up to four 256-bit passwords can be required) for individual blocks of flash memory
- Set levels of password-based read protection (up to four 256-bit passwords can be required) for up to five flash memory Read Locking Regions
- Set levels of password-based protection (up to four 256-bit passwords can be required) for the debug part

Table 94. PASS DCF records

DCF CS[14:0]	DCF address (binary)		DCF client description	Number of	
	[16:10]	[9:2]		Valid bits	DCF records
000_0000_0000_1000	0000000	00101000–00101011	Reserved		
000_0000_0000_1000	0000000	00101100	Censorship(1)	16	1

Table 94. PASS DCF records (continued)

DCF CS[14:0]	DCF address (binary)		DCF client description	Number of	
	[16:10]	[9:2]		Valid bits	DCF records
000_0000_0000_1000	0000000	00101101–00110111	Reserved		
000_0000_0000_1000	0000000	00111000	Fuse Bypass Enable		0
000_0000_0000_1000	0000000	00111001–00111111	Reserved		
000_0000_0000_1000	0000000	01000000	LOCK0_PG0(2)	32	1
000_0000_0000_1000	0000000	01000001	LOCK1_PG0(2)	32	1
000_0000_0000_1000	0000000	01000010	LOCK2_PG0(2)	32	1
000_0000_0000_1000	0000000	01000011	LOCK3_PG0(2)	32	1
000_0000_0000_1000	0000000	01000100	LOCK0_PG1(2)	32	1
000_0000_0000_1000	0000000	01000101	LOCK1_PG1(2)	32	1
000_0000_0000_1000	0000000	01000110	LOCK2_PG1(2)	32	1
000_0000_0000_1000	0000000	01000111	LOCK3_PG1(2)	32	1
000_0000_0000_1000	0000000	01001000	LOCK0_PG2(2)	32	1
000_0000_0000_1000	0000000	01001001	LOCK1_PG2(2)	32	1
000_0000_0000_1000	0000000	01001010	LOCK2_PG2(2)	32	1
000_0000_0000_1000	0000000	01001011	LOCK3_PG2(2)	32	1
000_0000_0000_1000	0000000	01001100	LOCK0_PG3(2)	32	1
000_0000_0000_1000	0000000	01001101	LOCK1_PG3(2)	32	1
000_0000_0000_1000	0000000	01001110	LOCK2_PG3(2)	32	1
000_0000_0000_1000	0000000	01001111	LOCK3_PG3(2)	32	1

1. See *Section 78.5.3: Censoring and uncensoring the device* for values.
2. See *Section 7.11.1.1: PASS_LOCKn_PGn register bit mapping* for values.

7.11.2 Tamper Detection Module (TDM) configuration

The Tamper Detection Module provides a type of flash memory erase protection mechanism that forces software to program a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased.

Additionally, the TDM provides a mechanism for configuring individual flash memory blocks as One Time Programmable (OTP), that is, write once, blocks.

The following sections provide details on the TDM configuration registers and DCF clients.

See the *Chapter 79: Tamper Detection Module (TDM)* for further details on the TDM, including the DCF clients and two additional memory-mapped registers.

7.11.2.1 Diary Base Address (DBA) DCF client

DBA DCF client holds the base address of the diary, which is the region of flash that contains records corresponding to erase operations. This DCF client is described in the TDM chapter.

7.11.2.2 Tamper Region Override (TO) DCF client

Tamper detection can be overridden by writing to enable bits (one per region) in the Tamper Region Override (TO) DCF client. This DCF client is described in the TDM chapter. The mapping is shown below.

DCF Client Address: 0x0001 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26. Tamper Region Override (TO) DCF client

The mapping of the Tamper Region Override Enable (TOE*n*) DCF client is as follows:

Table 95. TO[TOE*n*] field mapping

Field	TDR
TOE0	0
TOE1	1
TOE2	2
TOE3	3
TOE4	4
TOE5	5

This DCF client is more fully described in the TDM chapter

7.11.2.3 OTPEN*n* DCF clients

Configuring individual flash blocks as OTP blocks is accomplished by setting the appropriate values in the TDM_OTPEN*n* DCF clients.Each mapped bit is associated with a specific flash block. A “1” bit value indicates a block is OTP.

The mapping is shown in *Figure 27* - *Figure 30*.

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TSLOCK	Reserved	LOWLOCK[13:0]													
Flash Block Name	16 KB UTEST NVM Block Space	—	—	—	—	—	—	—	128 KB HSM Code block0	32 KB Code Flash block2	32 KB Code Flash block1	32 KB Code Flash block0	16 KB HSM Code block3	16 KB Code Flash block2	16 KB Code Flash block1	16 KB BAF block0

Bit Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	MIDLOCK[15:0]															
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB HSM EEPROM block1	16 KB HSM EEPROM block0

Figure 27. TDR*x*_LOCK0/OTPEN0 DCF client

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved															
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Bit Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	HIGHLOCK[15:0]															
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	16 KB FMC EEPROM Block3	16 KB FMC EEPROM Block2	16 KB FMC EEPROM Block1	16 KB FMC EEPROM Block0

Figure 28. TDR*x*_LOCK1/OTPEN1 DCF client

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit Name	A256KLOCK[31:16]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Bit Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bit Name	A256KLOCK[15:0]															
Flash block Name	—	—	—	—	—	—	—	—	256 KB Code Flash block6	256 KB Code Flash block5	256 KB Code Flash block4	256 KB Code Flash block3	256 KB Code Flash block2	256 KB Code Flash block1	256 KB Code Flash block0	128 KB Code Flash block0

Figure 29. TDR*x*_LOCK2/OTPEN2 DCF client

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit Name	A256KLOCK[63:48]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Bit Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bit Name	A256KLOCK[47:32]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 30. TDR*x*_LOCK3/OTPEN3 DCF client

7.11.2.4 TDRx_LOCKn DCF clients

Configuration of the 6 tamper regions is accomplished by setting field values in LOCK DCF clients (4 for each TDR).

Each set of 4 LOCKn DCF clients acts as a map of the MCU's flash memory blocks. A "1" in a LOCKn bit indicates that software must write a record before the corresponding block can be modified. The mapping of TDM_LOCKn DCF client bits to flash blocks is identical to the OTPEN*n* mapping and is shown in *Figure 27* - *Figure 30*. Any available flash block can be mapped to any tamper protection region.

7.11.2.5 DCF client details

Table 96 contains the DCF client information.

Table 96. Tamper Detection Module (TDM) DCF clients

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00000000	Diary Base Address	32	1
000_0000_0001_0000	0000000	00000001	Tamper Region Override	6	1
000_0000_0001_0000	0000000	00000010	Software Tamper Region Override Disable	6	1
000_0000_0001_0000	0000000	00000011 - 00000111	Reserved		
000_0000_0001_0000	0000000	00001000	OTPEN0(1)	10	1
000_0000_0001_0000	0000000	00001001	OTPEN1(1)	8	1
000_0000_0001_0000	0000000	00001010	OTPEN2(1)	9	1
000_0000_0001_0000	0000000	00001011	OTPEN3(2)	0	1
000_0000_0001_0000	0000000	00001100- 00010011	Reserved		
000_0000_0001_0000	0000000	00010100	TDR0_LOCK0	32	1
000_0000_0001_0000	0000000	00010101	TDR0_LOCK1	32	1
000_0000_0001_0000	0000000	00010110	TDR0_LOCK2	32	1
000_0000_0001_0000	0000000	00010111	TDR0_LOCK3	32	1
000_0000_0001_0000	0000000	00011000	TDR1_LOCK0	32	1
000_0000_0001_0000	0000000	00011001	TDR1_LOCK1	32	1
000_0000_0001_0000	0000000	00011010	TDR1_LOCK2	32	1
000_0000_0001_0000	0000000	00011011	TDR1_LOCK3	32	1
000_0000_0001_0000	0000000	00011100	TDR2_LOCK0	32	1
000_0000_0001_0000	0000000	00011101	TDR2_LOCK1	32	1

Table 96. Tamper Detection Module (TDM) DCF clients (continued)

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00011110	TDR2_LOCK2	32	1
000_0000_0001_0000	0000000	00011111	TDR2_LOCK3	32	1
000_0000_0001_0000	0000000	00100000	TDR3_LOCK0	32	1
000_0000_0001_0000	0000000	00100001	TDR3_LOCK1	32	1
000_0000_0001_0000	0000000	00100010	TDR3_LOCK2	32	1
000_0000_0001_0000	0000000	00100011	TDR3_LOCK3	32	1
000_0000_0001_0000	0000000	00100100	TDR4_LOCK0	32	1
000_0000_0001_0000	0000000	00100101	TDR4_LOCK1	32	1
000_0000_0001_0000	0000000	00100110	TDR4_LOCK2	32	1
000_0000_0001_0000	0000000	00100111	TDR4_LOCK3	32	1
000_0000_0001_0000	0000000	00101000	TDR5_LOCK0	32	1
000_0000_0001_0000	0000000	00101001	TDR5_LOCK1	32	1
000_0000_0001_0000	0000000	00101010	TDR5_LOCK2	32	1
000_0000_0001_0000	0000000	00101011	TDR5_LOCK3	32	1
000_0000_0001_0000	0000000	00101100 - 1111111	Reserved		

1. OTP DCF clients have the same structure and mapping as the write-protect bits of the flash and PASS module LOCK DCF clients. A value of ‘1’ in an OTP bit indicates a block is programmable only once. Note that a block cannot be reconfigured as non-OTP after it has been configured as OTP.
2. There are no flash memory blocks mapped to the OTPEN3 DCF client.

8 Reset and Boot

8.1 Introduction

This chapter describes the reset and boot phase, from the time a power on or reset is applied, till the completion of the Boot Assist Flash (BAF) code.

After reset, Core_2 is woken-up, HSM may be woken-up (refer to details in *Chapter 9: Device Configuration Format (DCF) Records*).

The following memory elements are involved in the boot-up process:

- UTEST flash memory
- TEST flash memory
- Boot Assist Flash (BAF)

8.1.1 TEST flash memory block

The TEST flash memory block contains Device Configuration Format (DCF) records used to hold trim values and other variables, as well as general device configuration information.

The trim values are for:

- Adjusting low-voltage and high-voltage detect circuit trip points
- Temperature sensor adjustments
- Power-on reset voltage trip point
- Analog-to-digital adjustments
- Internal RC oscillator (IRCOSC) trim values
- Power management configuration, for the devices supporting different regulation schemes

Note: The DCF records are written by the device manufacturer and programmed into TEST flash memory during production testing. Further programming of the TEST flash is disabled at the end of the factory test cycle.

8.1.2 UTEST flash memory block

The UTEST flash memory block also contains DCF records. Some UTEST DCF records are written by the factory and programmed during production testing. Others are written by the end user and programmed at the same time application code is programmed into the flash memory. UTEST DCF records are used to set up various control and configuration registers, including the Self-Test Control Unit (STCU2) and default memory configuration.

8.1.3 Boot Assist Flash

The Boot Assist Flash (BAF) contains factory code to facilitate the device boot-up procedure.

Refer to *Chapter 52: Boot Assist Flash (BAF)* for further details on this procedure.

8.2 Modules used in reset sequence

The modules involved in the SPC584Bx reset sequence are:

- Power Management Controller (PMC)
- Reset Generation Module (MC_RGM)
- Mode Entry Module (MC_ME)
- System Status and Configuration Module (SSCM)
- Self-Test Control Unit (STCU2)

8.2.1 Power Management Controller

The Power Management Controller (PMC) controls and monitors:

- Its own supply voltage
- The supply voltages to all the high- and low-voltage detect circuits
- The trip points for all the high- and low-voltage detect circuits
- The power supplies and reference voltages to the Analog-to-Digital Converter
- The major power supplies to the SPC584Bx

8.2.2 Reset Generation Module

The Reset Generation Module (MC_RGM) is a complex state machine that begins sequencing the SPC584Bx through the initial steps of the reset process. The MC_RGM does not execute program code, it is a state machine that centralizes the different reset sources and manages the reset sequence. Reset sources are organized into two categories: destructive and functional.

Refer to *Figure 31* for more information on the MC_RGM reset sequence.

8.2.2.1 Destructive resets

A destructive reset source is related to a critical error or dysfunction, usually caused by a hardware malfunction. When a destructive reset event occurs, software recovery is not possible and the contents of memory are assumed unknown. A full device reset sequence starting from PHASE0 is therefore applied to the device, ensuring a safe startup state for both digital and analog modules.

Examples of destructive resets are:

- Power-on reset
- Low voltage detection
- Reset escalation (when too many functional resets occur simultaneously)

8.2.2.2 Functional resets

A functional reset source is related to a less critical error or dysfunction usually not associated with hardware malfunction. When a functional reset event occurs, a partial reset sequence is applied to the device starting with PHASE1[FUNC]: most digital modules are reset normally and the state of analog modules, specific digital modules (for example, debug and flash memory modules) and system memory content is preserved.

Examples of functional resets are:

- External reset
- Machine check
- Software reset from mode entry
- Boundary scan instructions

8.2.3 Mode Entry module

The Mode Entry module (MC_ME) is responsible for configuring the execution mode, including the delivery of reset vectors to all the SPC584Bx cores.

8.2.4 System Status and Configuration Module

During the reset phase, the SSCM reads the DCF records in the TEST and UTEST flash memory areas in order to distribute within the device the pre-assigned configuration.

The DCF records are primarily for setting up the memory, configuring the Self-Test Control Unit (STCU2), and providing initial device configuration values.

The SSCM continues with the reset or boot-up sequence by locating boot vectors for the boot core and, if enabled for the HSM CPU. The start addresses are written into the Mode Entry Module (MC_ME) registers, which, in turn, will pass them to the proper CPU at the end of the reset phase.

8.2.5 Self-Test Control Unit

The Self-Test Control Unit (STCU2) is a self-contained module that runs a Memory Built-In Self-Test (MBIST). The DCF record can be used to select which individual tests are run for MBIST and disable tests for modules and memory elements that are not going to be used in user applications, thus shortening device boot-up time.

8.3 Reset sequence

The power-up reset sequence always begins with the application of power and follows different sequences depending on the condition of the SPC584Bx device and whether various modes are enabled from settings in the DCF records. For instance, the SPC584Bx device enters Serial Boot mode (the Serial Boot Loader receives startup code and begins program execution) if a valid boot header file is not found (conditioned by the Life Cycle value).

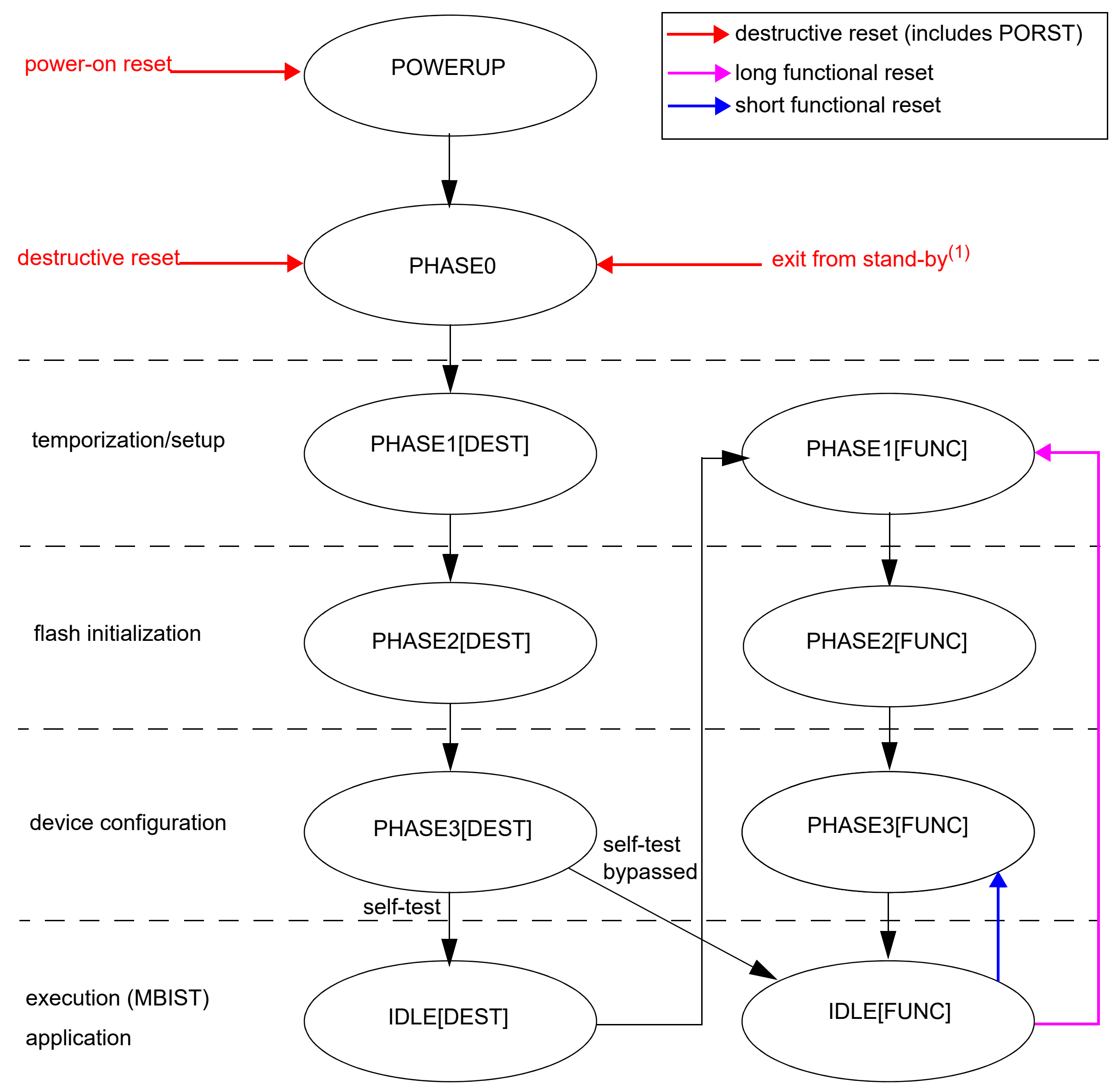
8.3.1 Power-on and the Reset Generation Module

When power is applied to the SPC584Bx, the Reset Generation Module (MC_RGM) advances the device through a series of steps shown in *Figure 31*.

The Reset Generation Module (MC_RGM) starts the System Status and Control Module (SSCM), which continues the boot-up process after the MC_RGM enters the IDLE state.

The STCU is reset on any power-on, destructive reset.

Figure 31. Reset Generation Module Reset Sequence



1. No self-tests are executed.

Table 97. Module status during reset phases

Module	PHASE								
	P0	P1 [DEST]	P2 [DEST]	P3 [DEST]	IDLE [DEST]	P1 [FUNC]	P2 [FUNC]	P3 [FUNC]	IDLE [FUNC]
IRCOSC	ON	ON	ON	ON	ON	ON	ON	ON	ON
PMC	ON	ON	ON	ON	ON	ON	ON	ON	ON
FUSE	ON	ON	ON	ON	ON	ON	ON	ON	ON
MC_RGM	ON(1)	ON	ON	ON	ON	ON	ON	ON	ON
SWT	RESET	RESET	ON	RESET	IDLE	RESET	ON	RESET	ON
Flash module	RESET	RESET	ON	ON	IDLE	RESET	ON	ON	ON
XOSC	RESET	RESET	RESET	ON(2)	ON(2)	ON(2)	ON(2)	ON(2)	ON(2)
DEVICE CONFIG	RESET	RESET	RESET	LOAD	HOLD	HOLD	HOLD	HOLD	HOLD
SSCM	RESET	RESET	RESET	ON	ON	RESET	RESET	ON	ON
FCCU	RESET	RESET	CONFIG	CONFIG	IDLE	RESET/ ON(3)	CONFIG/ ON	CONFIG/ ON	ON
STCU	RESET	RESET	CONFIG	CONFIG	ON	RESET/ IDLE(4)	CONFIG/ IDLE	CONFIG/ IDLE	ON/ IDLE(5)
MEMU	RESET	RESET	ON	ON	IDLE	RESET/ ON(6)	ON	ON	ON
PASS	RESET	ON	ON	ON	ON	ON	ON	ON	ON
HSM	RESET	RESET	RESET	RESET	IDLE	RESET	RESET	RESET	ON
Boot CPU	RESET	RESET	RESET	RESET	IDLE	RESET	RESET	RESET	ON
PIT	RESET	RESET	RESET	RESET	IDLE	IDLE/ON	IDLE/ON	IDLE/ON	IDLE/ON
Other modules including cores	RESET	RESET	RESET	RESET	IDLE	RESET	RESET	RESET	ON
								Short functional reset	
						functional reset			
	destructive reset								

1. RESET only during POR.
2. Depending on flash configuration.
3. FCCU remains ON during functional reset; it is instead reset in PHASE1[FUNC] if coming from IDLE[DEST].
4. The STCU remains idle during functional reset; it is instead reset on long external reset.
5. Can be triggered by SW. Please refer to *Chapter 76: Self-Test Control Unit (STCU2).*
6. MEMU is reset whenever STCU is reset.

8.3.2 Power-up phase: power stabilization

When a power-on reset event occurs (initial application of power), the reset generation module (MC_RGM) causes the device to enter the power-up phase of the reset sequence (refer to *Figure 31*). The only method to enter the power-up phase is from a power-on reset event.

The device remains in the power-up phase until the Power Management Controller module (PMC) determines that all power supplies are in their respective operational ranges.

The supplies for initial configuration are:

- VDD_LV: low voltage supply for digital module.
- VDD_HV_FLA: high voltage supply for flash module.
- VDD_LV_FLA: low voltage supply for flash module.
- VDD_HV_IO_MAIN: high voltage supply associated with system pins (PORST, TEST Mode), including power management unit.

Peripheral supplies such as , VDD_HV_IO_ETH or VDD_HV_ADV do not gate the power-up process. The low- and high-voltage detect circuits for these and other power supplies are managed by the PMC, which provides enable and status bits for these voltage detection circuits.

The PMC sends a signal to the Reset Generation Module when power stabilization has been achieved. The Reset Generation Module then advances to the PHASE0 state (refer to *Figure 31*). During the power-up phase, the PMC does the following:

- Drives to a logic ‘0’ all VDs (voltage detectors) signals monitored by the Power Management Control (PMC) module.
- Monitors its own power supply voltage (VDD_HV_PMC) to determine that the applied voltage is within a specified range.
- Monitors the core power supply voltage (VDD_LV_CORE) to determine that the applied voltage is within a specified range.
- Monitors the low- and high-voltage detect circuit power supply voltage to determine that the applied voltage is within a specified range.
- Holds the outputs of the internal low- and high-voltage detect circuits at a ground state until the PMC determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Enables all low- and high-voltage detect circuits once all power supplies are at their functional levels.
- Begins to monitor the various power supplies using the high- and low-voltage detect circuits.
- Provides status information to the Reset Generation Module (MC_RGM) indicating whether or not power is properly applied.

To exit power-up and enter PHASE0:

- All enabled destructive resets must be processed.
- All power supplies must be at their functional voltages.
- The PMC must signal the MC_RGM that all power supplies are at the required levels.

8.3.3 PHASE0 Phase: analog supply initial configuration

This phase is entered:

- On exit from power-up phase.
- On PORST pin falling edge detection except for the initial power-up sequence.
- Immediately from PHASE2[DEST], PHASE3[DEST], IDLE[DEST], PHASE1[FUNC], PHASE2[FUNC], PHASE3[FUNC] or IDLE[FUNC], on the occurrence of a PORST falling edge event other than power-on reset.

During PHASE0:

- All digital modules are reset, including safety, security, and test modules.
- All trimming bits for analog modules are reset.
- Startup of the internal analog modules (IRCOSC, FUSE, Flash memory, and I/Os) begins:
  - PMC determines that proper voltages are applied.
  - I/O pins are ensured correctly configured: outputs are driven to known levels, inputs ignored.

To exit PHASE0 and enter PHASE1[DEST]:

- All enabled destructive resets must be processed.
- Gating PHASE0 must be released.
- All processes initiated in PHASE0 must be completed. This generally includes startup of the internal analog modules: PMC, IRCOSC, and I/Os.

Note: The core voltage must rise above upper threshold to exit PHASE0. This ensures significant hysteresis on low voltage supply avoiding spurious low voltage detection during the power-up process.

8.3.4 PHASE1[DEST] Phase: temporization and monitoring setup

This phase is entered on exit from PHASE0.

During PHASE1[DEST]:

- Initial configuration of the watchdog to monitor the reset sequence
- Temporization for reset signal propagation in case of reset

Test modules are available during PHASE1[DEST], but their functionality is limited by the security module.

All PHASE1[DEST] tasks must be completed before entering PHASE2[DEST].

8.3.5 PHASE2[DEST] Phase: flash initial configuration

This phase is entered on exit from PHASE1[DEST].

During PHASE2[DEST]:

- Reset is released to the flash memory module by the Reset Generation Module.
- The flash memory module starts its initial configuration process.
- The flash memory initialization is performed by a state machine internal to the flash module.
- The SWT starts monitoring flash memory configuration execution so that a reset occurs in the event of configuration failure.

Note: SWT is hardware configured to generate destructive reset in case of flash memory failure during flash boot process. In this case the software can read the cause of the reset by the MC_RGM_DES.F_FIF flag. On boot completion, the watchdog event is converted to generate an FCCU fault only.

All PHASE2[DEST] tasks must be completed before entering PHASE3[DEST].

8.3.6 PHASE3[DEST] Phase: device configuration

This phase is entered on exit from PHASE2[DEST].

During PHASE3[DEST], the System Status and Configuration Module (SSCM) starts by retrieving the Device Configuration Format (DCF) record from the TEST and UTEST flash memory areas, and uses the configuration and initialization information therein to:

- Configure the Self-Test Control Unit (STCU2) and the initial memory map of the device.
- Transfer reset vectors for the various cores to the Mode Entry Module (MC_ME). This information enables tests that the STCU runs and also provides the memory map setup information so that the STCU can test memory.
- Write trim values for analog modules including the Analog-to-Digital Converter, low- and high-voltage detect circuits, and the temperature sensor to their respective registers.

The DCF record programmed into the TEST flash memory area is written by the manufacturer. The portion of the DCF record programmed into the UTEST flash memory is usually written by the customer, although there is a default DCF record programmed into both the TEST and the UTEST flash memory areas during device production testing.

The DCF record also provides configuration information for the following:

- Trimming of the analog modules (for example, voltage regulator, voltage detectors, I/Os, and oscillator)
  - Trim values are in DCF records in the TEST flash memory area
  - Trim values are determined during production test and programmed into the TEST flash memory area
  - Security configuration
- Application configuration bits:
  - Watchdog configuration
  - Safety execution directives (self-test, MBIST)
  - Boot Assist Flash options
  - Oscillator startup during reset sequence configuration

To exit PHASE3[DEST]: All processes that need to be completed in PHASE3[DEST] must be finished.

This phase depends on actual trim values, especially those used for safety execution directives. This device configuration is maintained until the next internal power-on reset. Each time a destructive reset event occurs, the SSCM takes values from the DCF record and writes them to the applicable registers.

During this phase, the internal STCU watchdog is started and is only cleared after completion of STCU configuration.

If self-test execution is the configuration information from the DCF record requests, the device moves to IDLE[DEST] and the STCU2 begins self-test. At the end of the test, the

STCU2 asserts a functional interrupt causing the Reset Generation Module to move to PHASE1[FUNC]. If self-test execution is not requested, the device moves directly to IDLE[FUNC]. The Reset Generation Module determines whether self-test execution is enabled or not by examining an entry in the DCF record stored in the UTEST flash memory.

Note: When self-test execution is not requested, it is possible to maintain the device in PHASE3[DEST] by forcing the PORST pin low. This can be used for debugging purposes by allowing external test/development equipment to connect through JTAG to the internal debugger.

8.3.7 IDLE[DEST] Phase: self-test execution

This phase is entered upon exit from PHASE3[DEST] if the execution of self-test is enabled.

During the IDLE[DEST] phase:

- Dedicated watchdog is configured to monitor self-test execution completion within expected time. The length of this phase is variable depending on the self-test requirements.
- The Self-Test Control Unit (STCU2) executes all tests specified by the DCF information retrieved in PHASE3[DEST].
- The self-test engine updates the self-test completion flag and triggers an associated functional reset on completion of self-test.

The length of this phase is variable depending on the self-test requirements.

Note: When the SSCM is decoding the Device Configuration field, it sets and clears various flags in STCU registers and memory that can later be examined by the SSCM to determine which self-test directives must be run.

Exiting IDLE[DEST] and moving to PHASE1[FUNC]:

- The IDLE[DEST] phase is automatically exited when the self-test code generates a functional reset on completion.
- The functional reset causes the reset generation module to move to the PHASE1[FUNC] phase.

Note: In the case of unrecoverable fault detected during the execution of the offline BIST, the STCU requests a destructive reset and device re-enters PHASE0[DEST].

In the case of a permanent fault, when the reset escalation is reached, the hardware keeps the device in PHASE0[DEST] until a Power On Reset occurs.

Whereas in the case of a transient fault, the software can read the cause of the reset by reading the MC_RGM_DES.F_SUF flag.

8.3.8 PHASE1[FUNC] Phase: temporization and monitoring setup

This phase is entered either:

- On exit from IDLE[DEST].
- Immediately from PHASE2[FUNC], PHASE3[FUNC], or on IDLE[FUNC] when a non-masked external or functional reset event occurs, providing the source of the reset event has not been configured to trigger a ‘short’ reset sequence.

During PHASE1[FUNC]:

- Initial configuration of the watchdog to monitor reset process.
- Temporization for reset signal propagation in case of reset.

To exit PHASE1[FUNC] and enter PHASE2 [FUNC]:

- All enabled, non-shortened functional resets must be processed.
- All processes started in PHASE1[FUNC] must be completed.
  - Initial configuration of the watchdog to monitor reset process.
  - Temporization for reset signal propagation in case of reset.

During this phase, test modules are available.

8.3.9 PHASE2[FUNC] Phase: flash initial configuration

This phase is entered on exit from PHASE1[FUNC].

During the PHASE2[FUNC] phase:

- The reset signal is released to the flash memory.
- Flash initialization and configuration begins using a state machine that is internal to the flash memory module.
- A watchdog monitors flash initialization to ensure a reset in case of flash memory configuration failure.

The reset state machine exits PHASE2[FUNC] and enters PHASE3[FUNC] on verification that all processes started in PHASE2[FUNC] are completed.

8.3.10 PHASE3[FUNC] Phase: device configuration monitoring

PHASE3[FUNC] is entered:

- On exit from PHASE2[FUNC].
- Immediately from IDLE[FUNC] when an enabled, short functional reset event occurs.

During PHASE3[FUNC], configuration information contained within the UTEST flash memory sector is checked against information extracted during PHASE3[DEST] and written to the Self-Test Control Unit (STCU2). In case of mismatches, a dedicated boot fault is triggered within the FCCU which, if enabled, can cause a system reset. The purpose of PHASE3[FUNC] is to verify that the DCF information transferred to the STCU, MC_ME and other modules, in PHASE3[DEST], was performed correctly and has not been corrupted.

The Reset Generation Module exits PHASE3[FUNC] and enters the IDLE[FUNC] on verification of the following:

- All processes started in PHASE3[FUNC] are completed.
- PORST pin is not forced low externally.
- A minimum number of cycles have elapsed since the last enabled, shortened functional reset event.

After all PHASE3[FUNC] internal actions have been completed, it is possible to maintain the device within PHASE3[FUNC] by forcing the PORST pin low. This can be used for debugging purposes in order to connect through the JTAG port to the internal debugger.

8.3.11 IDLE[FUNC] Phase

This is the Reset Generation Module's final phase of the reset sequence. It is entered either on exit from PHASE3[DEST] (if self-test was not enabled) or at the completion of PHASE3[FUNC]. When the IDLE[FUNC] phase is reached, the MC_RGM releases control of the system to the System Status and Control Module. The SSCM, which started in

PHASE3[FUNC], (or in PHASE3[DEST] in case no self-test was enabled) runs and waits for a signal from the MC_RGM indicating that the MC_RGM is now in the IDLE state. The MC_RGM then waits for new reset events that can trigger a reset sequence. The SSCM continues with the system boot-up sequence.

8.3.12 System start-up

At the start-up, only one processor (from now on called Boot CPU) and, if available and enabled, the Security Module CPU wake up.

8.3.12.1 Boot CPU reset vector

After the reset phase is completed, the Boot CPU starts executing at the value forced onto its reset vector. The reset vector is driven by the ME peripheral, according to the value stored into the CADDR register.

The value of the CADDR register depends on whether the reset is generated as a consequence of a wake-up reset event (restart from Stand-by) or any other event. The Flash status (powered on or off) also concurs to select the reset vector value.

In case the reset is not due to a wake-up event, the CADDR registers for the Boot CPU are written by the SSCM, according to the system startup configuration, which is stored in Flash and read by the SSCM from the Flash during the reset.

In case the reset is due to a wake-up event, the ME reads the CADDR reset vectors from the RGM, which had saved a copy of the CADDR registers in its internal latches, which resides in the low-power domain.

8.3.12.2 Boot CPU start-up

The Boot CPU startup configuration comprises of the following settings:

- BAF configuration DCF. It is used to decide whether the BAF code (stored by the manufacturer in the Flash BAF block) will be skipped or executed. The BAF start address is also set within the same DCF.
- Application boot records (programmed by the customer). It is used to set the customer application entry point after reset or after the BAF execution (if enabled).

The target for this device is to let the customer choose whether either the BAF code shall be started after the reset phase or directly the customer application, using the same sales part number.

The BAF configuration DCF shall be permanently written into the UTest block.

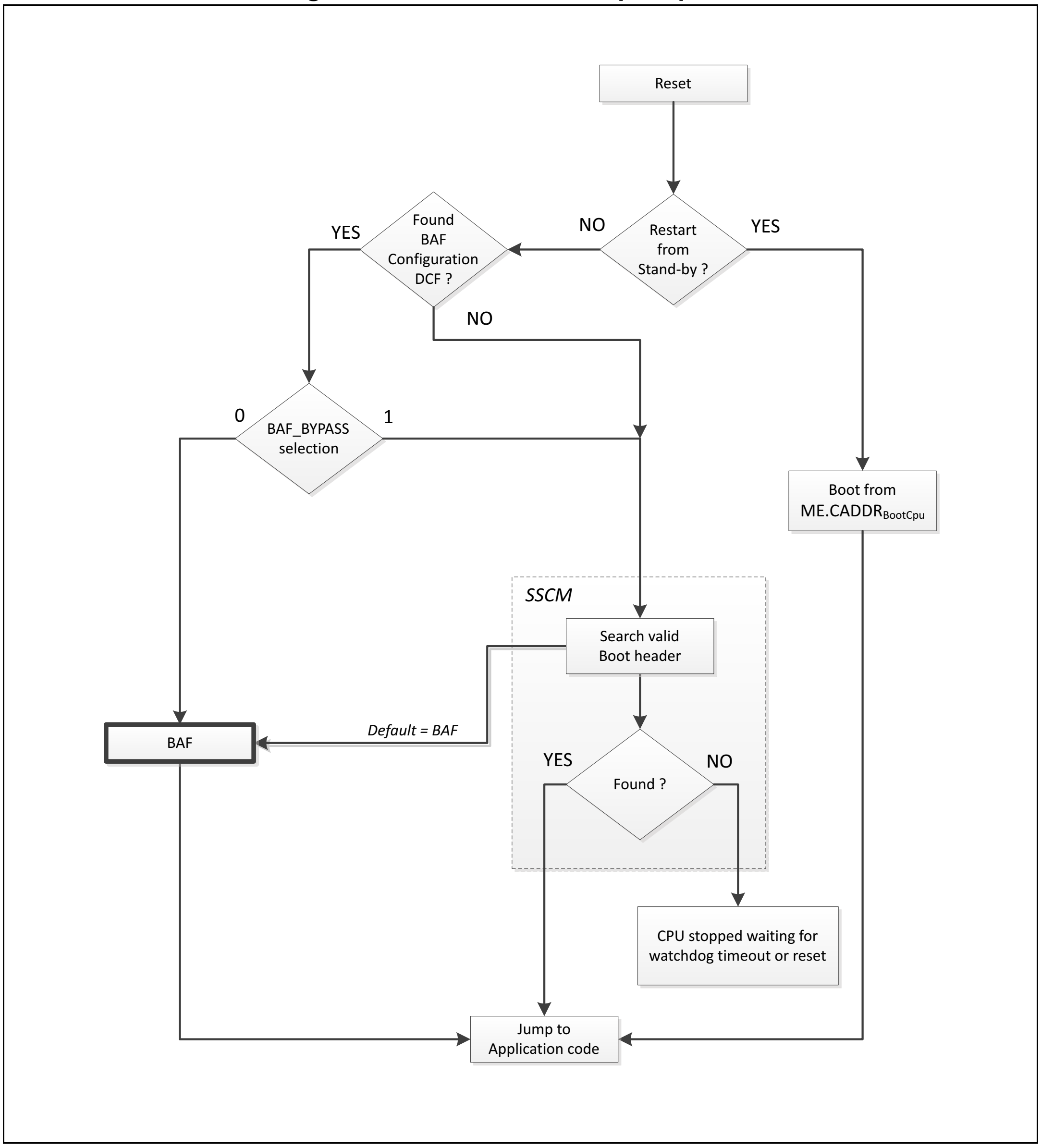
If the BAF configuration DCF is not programmed, the default value for the reset start address depends on the Life Cycle.

If the BAF configuration DCF is programmed, its value determines whether the CPU will start directly from the BAF code (programmed by ST in the Flash BAF block) or at the application entry point defined by the customer.

The application entry point can be retrieved both in HW (by the SSCM or ME, depending on the fact the system restarts from Stand-by or not (refer to *Section 8.3.12.1*) or in SW (by the BAF). In both cases, the application entry point is retrieved by searching for a valid Boot Record, which shall be programmed by the customer in one of the pre-determined Flash memory locations looked at by the SSCM or the BAF code. Next sections provide more details about the Boot Record structure and search operation.

8.3.12.2.1 Start-up flow chart

Figure 32. Boot CPU start-up sequence



8.3.12.2.2 HW boot record search (SSCM search)

If the programmed value of the BAF configuration DCF skips the BAF, then the SSCM module looks for a valid software entry point, called SSCM boot record. The SSCM boot record is organized as shown in the *Table 98*. The “Application start address” value will be passed to the CADDR register of MC_ME, which is used to set the CPU start address (reset vector) after the reset phase.

Table 98. Boot record structure searched by the SSCM

Offset	Field description	
0x4	Application start address	
0x0	SSCM Boot Record Tag (0x5A)	Reserved

The SSCM searches for a valid boot record at the addresses specified in *Table 99*. A boot record is considered valid if it contains the value 0x5A in the "Boot Record Tag" field.

The "Search Order" column specifies which location is searched first.
The SSCM stops the search as soon as a valid boot record is found.

Table 99. SSCM boot record search locations

Search order	Address	Location	SSCM configuration parameter
0 (First)	0x00FC_0000	Flash 16 KB Code Block 1	RCHW_LOC_0
1	0x00FC_4000	Flash 16 KB Code Block 2	RCHW_LOC_1
2	0x00FC_8000	Flash 32 KB Code Block 3	RCHW_LOC_2
3	0x00FD_0000	Flash 32 KB Code Block 4	RCHW_LOC_3
4	0x00FE_0000	Flash 128 KB Code Block 0	RCHW_LOC_4
5	0x0100_0000	Flash 256 KB Code Block 1	RCHW_LOC_5
6 (Last)	0x0040_4000	Flash 16 KB BAF block	RCHW_LOC_6

Depending upon the fact that a valid SSCM boot record is found and the content of the SSCM boot record, three scenarios can happen:

1. If a valid Boot Record is programmed by the customer, then the application is executed after the reset from the specified entry point address.
2. If no valid Boot Record is programmed by the customer, then the default BAF Boot Record is retrieved and the BAF code is executed after the reset.
3. If no valid Boot Record is found, then the BAR code is executed after the reset. This situation happens only during ST production stage.

Note: When the device is shipped to the customer, the slot number 6 is programmed with a valid SSCM Boot Record, which points to the BAF code. This is the default start address for the SSCM Boot CPU when no other valid Boot Record are programmed or found. The content of the slot number 6 cannot be changed.

8.3.12.2.3 SW boot record search (BAF search)

The BAF is executed if:

- the BAF configuration DCF does not bypass the BAF code.
- the BAF configuration DCF does bypass the BAF code and no valid Boot Record is programmed by the customer.

The latter case happens because before the unit is delivered to the customer a valid SSCM boot record pointing to the BAF code is programmed at the location number 6. This boot record will point to the BAF entry point. Because the boot record number 6 is the last one being searched by the SSCM, this means that, if no other boot record is programmed, the software will start from the BAF. Notice that the location of the boot record number 6 is placed within the BAF block itself, making it not erasable and not over-programmable (the BAF block is OTP).

When the BAF code is executed after the reset sequence is completed, a valid application Boot Record for the application entry point is searched. The mechanism is the same executed by the SSCM: a set of fixed locations are analyzed and, if a valid boot record is found, then, at the end of the BAF code, the CPU will start executing from the application entry point.

Note that the valid boot record is identified by a different tag: 5Ah for the SSCM; A5H for the BAF. This will make it possible to program the same boot record locations and launch the application directly after the reset or after the BAF code execution. *Table 100* shows the BAF boot record structure.*Table 102* shows the addresses of the location searched by the BAF code. *Figure 33* shows a flow chart of the whole reset sequence, including the BAF search for a valid boot header.

Table 100. Boot record structure searched by the BAF

Offset	Field description	
0x4	Boot CPU (CPU_2) Reset vector	
0x0	BAF Boot Record Tag (0xA5)	CPU enable flags (refer to *Table 101*)

Table 101. CPU enable flags

16-19	20-23	24-27	28	29	30	31
0000	0000	0000	0	0	0	CPU_2

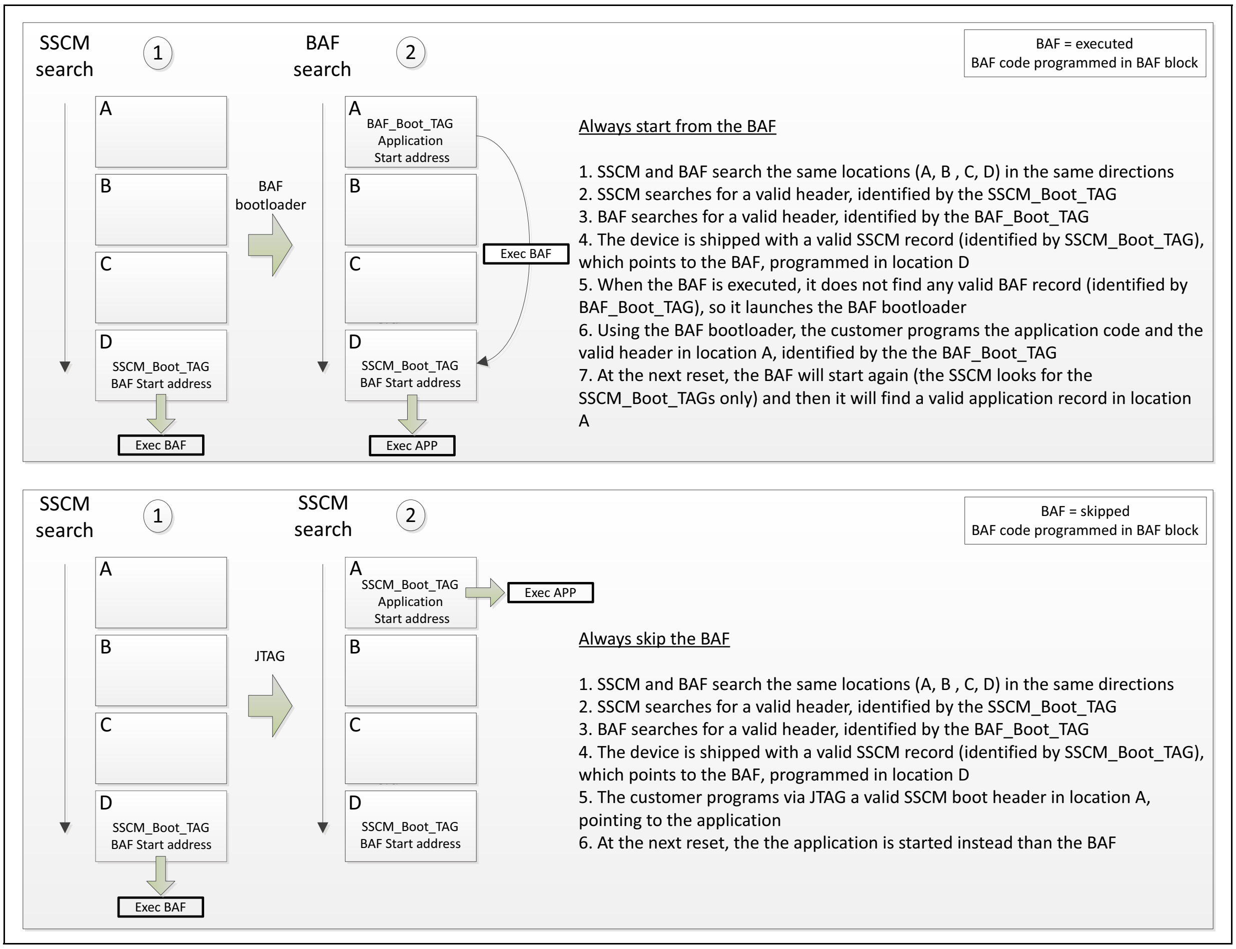
Table 102. BAF boot record search locations

Search order	Address	Location
0 (First)	0x00FC_0000	Flash 16 KB Code Block 1
1	0x00FC_4000	Flash 16 KB Code Block 2
2	0x00FC_8000	Flash 32 KB Code Block 3
3	0x00FD_0000	Flash 32 KB Code Block 4
4	0x00FE_0000	Flash 128 KB Code Block 0
5 (Last)	0x0100_0000	Flash 256 KB Code Block 1

8.3.12.2.4 HW and SW boot header search summary

The following picture summarizes the mechanism adopted to search for a valid boot header with the SCCM or the BAF code, or both.

Figure 33. Valid boot header search



Note: for sake of simplicity, the picture shows only 4 boot record positions.

8.3.12.3 Security Module CPU start-up

The Security Module CPU is also woken up after a reset. The Security Module CPU receives the start address as described in *Section 8.3.12.2.2: HW boot record search (SSCM search)*.

The SSCM looks for a valid boot header record and then direct the Security Module CPU to the application start address specified within the record.

8.3.12.3.1 HSM restart from Stand-by

In case of restart from stand-by, then the application entry point is conditioned by the Flash status (powered on or off) and by two DCF:

1. a DCF (to be placed in UTest under the customer control) determines whether the security module should also restart after Stand-by[c].

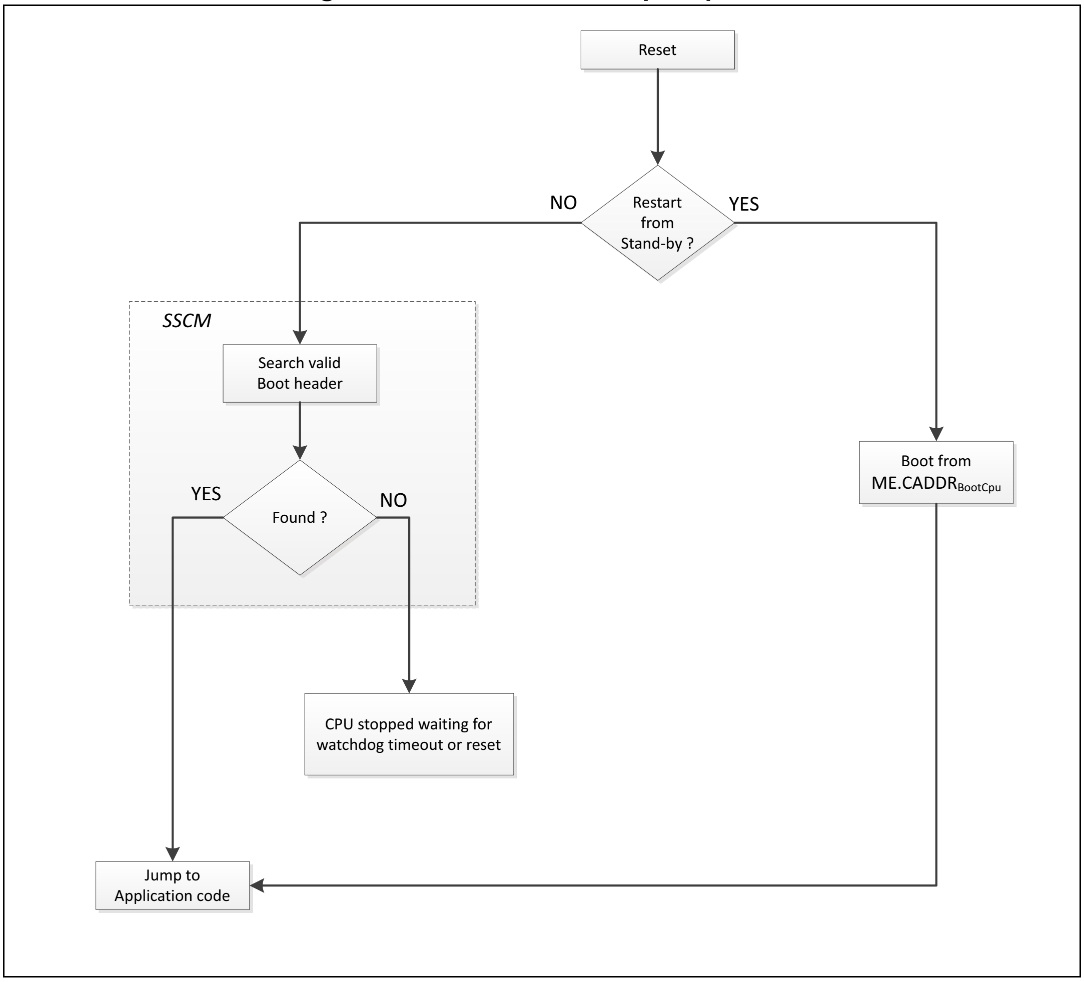
---

c. If the HSM is flagged by this DCF to not restart, then also the HSM Standby RAM can be turned off during Stand-by

2. a DCF (to be placed in UTest under the customer control) is used to inhibit the possibility to let the Flash powered off after a restart from Stand-by. If the restart is from Stand-by, the Flash is powered off, and the DCF client allows this scenario, then the Boot CPU restarts from the address stored into the corresponding ME.CADDR register. Otherwise, the Boot CPU restarts following the same flow as per a non stand-by reset.

8.3.12.3.2 Start-up flow chart

Figure 34. HSM CPU start-up sequence



8.3.12.3.3 Boot vector search

The Security Module start vector is searched by the SSCM in a similar way as described in the chapter for the Boot CPU (HW SSCM search).

The SSCM stops the search as soon as a valid boot record is found within one of these locations.

Table 103. SSCM boot record search locations

Search order	Address	Location
0 (First)	0x0060_C000	Flash 16 KB Code Block0
1	0x0061_0000	Flash 128 KB Code Block0

8.3.12.4 Restart from Stand-by RAM with the Flash powered off

This feature makes it possible to wake-up the micro-controller from the Stand-By mode and let the Boot CPU restart directly from the RAM, while the Flash is powered off.

Note: For security reason, the possibility to keep the Flash powered off in any execution modes shall be gated by a DCF.

In case of restart from stand-by with the Flash powered-off, only the Boot CPU shall be woken up. Other CPUs shall be kept under reset. The restart after Stand-by from RAM is made possible by using the Mode Entry registers:

- ME_CCTL*n* => to set up the processor status (ON/OFF) for each mode.
- ME_CADDR*n* => to set up the processor reset vector.
- ME_<mode>_MC => to set up whether the Flash is ON or OFF.

Note: The registers related to the HSM CPU can be modified by the HSM only.

8.3.12.5 Start-up summary

Table 104. Start-up summary

Start from	Flash status	Security Module CPU (1)	Boot CPU	Other CPUs (2)
Power-on / Destructive reset	ON	It starts from the address provided by SSCM	Depending on the BAF configuration DCF setting, it starts from the BAF or from the address provided by SSCM	They can be woken-up with a ME mode change
Functional reset	ON	It starts from the address provided by SSCM	Depending on the BAF configuration DCF setting, it starts from the BAF or from the address provided by SSCM	They can be woken-up with a ME mode change

1. If existing and enabled.
2. For MCUs with multiple application processors.

8.3.13 Waking-up other CPUs

The process of waking-up the other CPU is handled by writing some registers within the ME module and triggering a Mode Change.

If the BAF code is executed, it reads form the Boot Header record the information regarding which CPU shall be woken up and at which address shall they start from, and then applied a Mode change to enable them.

If the BAF code is not executed, it is responsibility of the user code to properly write the ME module registers and trigger a mode change in order to wake up the other CPUs form the intended addresses.

8.3.14 BAF Bootloader

The BAF code includes a way to upload via CAN or UART some code into the system RAM and execute it. The possibility to execute the Bootloader is conditioned by the Life Cycle status, which regulates what can or cannot be done according to the stage of the micro-controller. If the Bootloader is allowed by the Life Cycle, then it is executed whenever the Boot CPU, executing code in the BAF, does not find a valid boot header file in flash memory. This mode uses a defined protocol to receive a program over a serial port via LINFlexD or CAN. The Boot CPU then jumps to the first instruction of the uploaded program and begins execution.

The BAF bootloader code includes the following steps:

1. Configure LINFlexD or M_CAN module and external pins
2. Wait for START WORD
3. Receive START WORD
4. Receive START ADDRESS
5. Receive DOWNLOAD SIZE
6. Receive CODE
7. Branch to START ADDRESS

Note: There is a software watchdog timer that is started by the BAF code. If a program is not received before the watchdog timer timeout, a reset occurs.

9 Device Configuration Format (DCF) Records

9.1 Introduction

Device Configuration Format (DCF) records are used to configure certain registers in the device during system boot while the reset signal is asserted. An individual DCF record is 64 bits long. It consists of a pointer to the location of a register internal to the device and the respective data.

Note: Because the Flash memory programming resolution is 128 bits then 2 DCF records must be written within 1 Flash programming operation.

There are two broad categories of DCF records: TEST DCF Records and UTEST DCF Records.

- TEST DCF records are programmed by the MCU manufacturer. They are used mainly to:
  - Program registers involved in trimming trip points for voltage comparators
  - Adjusting analog to digital voltage supplies
  - Trim oscillator frequencies
  - Enable RAM repair

Note: The TEST DCF Records are programmed into TEST Flash during production and cannot be modified. TEST Flash is not visible to the user.

- UTEST DCF records can be either:
  - Factory programmed during production testing (see “UTEST Flash memory map” table in the Memory Map chapter).
  - User programmed when application code is written into the Flash memory. User-supplied UTEST DCF records start at the next location in the UTEST memory map following the factory UTEST DCF records. The user-defined DCF records set up the initial memory map, define which tests the Self-Test Control Unit (STCU2) runs during the boot sequence, enable the HSM, assign Flash memory blocks to specific tamper detect regions, assign memory blocks as OTP and assign Flash blocks to be associated with specific password groups.

Note: DCF records are HW processed by SSCM during reset.

Soft DCF records are SW processed by BAF code execution. Please see Section 52.3.5.1: ‘Soft’ DCF clients in *Chapter 52: Boot Assist Flash (BAF).*

System boot is a complex process requiring a considerable amount of initialization to take place before releasing reset. Before the device can be used in an application, the user application code, reset vectors for all of the CPUs and the DCF records must be properly programmed into their respective Flash memories.

9.2 DCF clients

DCF clients are 32-bit hardware registers inside a module that receives and stores the data from a DCF record. This stored data is used to initialize registers and to configure features. DCF Clients have a default value before any DCF Records are written and may have special writing constraints like ‘Write Once’ or only allowing bits to be written from ‘1’ to ‘0’ or vice versa.

A list of DCF clients is shown in *Table 107*. (TEST DCF Record Only clients are not shown as these are not accessible by the user).

The DCF Records select the target DCF client via a 30-bit field in the DCF Record consisting of:

- A 15-bit Chip-Select field: each module that includes DCF clients is assigned a Chip Select during chip definition.
- A 15-bit Address field: the address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

9.3 DCF records

DCF records appear as contiguous double-word (64 bits) entries programmed in a reserved area of OTP UTEST Flash memory beginning at 0x0040_0300. Take care to write 128 bits which is the ECC granularity (complete with a dummy DCF if needed). Address alignment is also on 128 bits.

Warning: Over Program Protection will generate an unrecoverable ECC error if only one DCF record (64 bits) is written.

Caution: Once a stop record is detected, the SSCM will stop scanning for further records.

The structure of a DCF Record is shown in *Figure 35*.

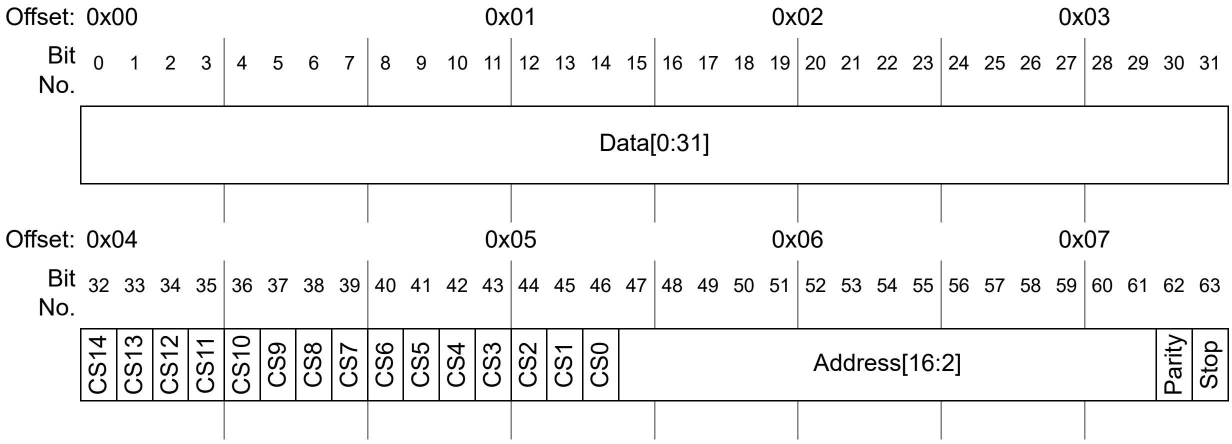


Figure 35. DCF record structure

Table 105. DCF record field description

Field	Description
0:31 Data[0:31]	32 bits of data to be written to the DCF Client.
32:46 CS*n*	Chip Select n. **Note:** Only assert one chip per DCF record to select the target module for the DCF client. All other Chip Selects should be negated. 1 Chip select is asserted. 0 Chip Select is negated.
47:61 Address[16:2]	Address of the DCF client within the selected module. **Note:** Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF Client addresses are defined in each module chapter.
62 Parity	Parity Bit for the DCF Record. **Note:** This bit is NOT implemented for DCF Clients written from UTEST.
63 Stop	This bit indicates the end of the list of DCF Records. **Note:** The Erased state of Flash is 0xFFFF_FFFF_FFFF_FFFF. Therefore the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list. 1 End of the list. 0 NOT the end of the list.

The following structure must be present:

- The first record must be a start record.
- DCF records containing configuration data must immediately follow the start record with no blank records between an unprogrammed record is interpreted as a stop record and no DCF records following that record are processed.
- The end of the configuration records are indicated by the presence of a bit in the DCF record (stop record).

Each record is 64 bits long. *Figure 36* shows the DCF start record.

Caution: The start record must be placed at the beginning of the DCF area in UTEST Flash memory to indicate that the following data records must be processed.

Figure 36. DCF start record

0x00 0:31	0x04 32:63
0x05AA55AF	0x00000000

During system boot, while the reset signal is asserted, the SSCM reads the device configuration records and writes the data to the appropriate internal module registers. Many operational aspects of the device are therefore already configured when the reset signal is released and normal device operation begins.

Some device configuration records are calculated during production testing and programmed into the TEST Flash memory area. Other DCF records are developed by the user and programmed into the UTEST Flash area.

The stop bit designates the last device configuration record in a set of records. If it is set to ‘1’ within a DCF, then this DCF is the last one being read during the reset phase. A DCF can have the stop bit set to ‘1’ in two modes:

a) The Flash memory location of a DCF is not programmed (that is all the bits are set to ‘1’).

b) A DCF is programmed and the stop bit is set to ‘1’.

When the device is shipped from the factors, some DCF records are already programmed and the Flash memory location after the last DCF is left unprogrammed to make it possible to add additional DCF records.

Due to the Flash programming memory resolution, which is 2-times the length of a DCF record, in case an odd number of DCF shall be written but additional DCF should be programmed later on, then a dummy DCF should be written to pad the last 64 bits of the programming data.

A dummy DCF can be one of the following:

a) The same DCF record written within the 128 bits (that is the DCF record is written two times).

b) A DCF record with an invalid Chips Select (refer to the *Table 107* for a list of used Chip Selects).

The padded DCF must have the stop bit set to ‘0’ in order to be able to add new DCF records.

The general format of the stop record is shown in *Figure 37*. Only the stop bit needs to be ‘1’ in order to form a stop record, all other bits are ignored. An unprogrammed location in UTEST Flash is interpreted as a stop record.

Figure 37. DCF stop record

0:31	32:62	63
Reserved	Reserved	1

If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 106*.

Table 106. Series of DCF records in UTEST Flash memory

<table>
<tr><th>ADDR offset</th><th colspan="4">DATA</th></tr>
<tr><td>0x00</td><td colspan="4">0x05AA55AF</td></tr>
<tr><td>0x04</td><td colspan="3">0x00000000</td><td>STOP=0</td></tr>
<tr><td>0x08</td><td colspan="4">WDATA[31:0]</td></tr>
<tr><td>0x0C</td><td>CS[14:0]</td><td>ADDR[16:2]</td><td>PRTY</td><td>STOP=0</td></tr>
<tr><td>0x10</td><td colspan="4">WDATA[31:0]</td></tr>
</table>

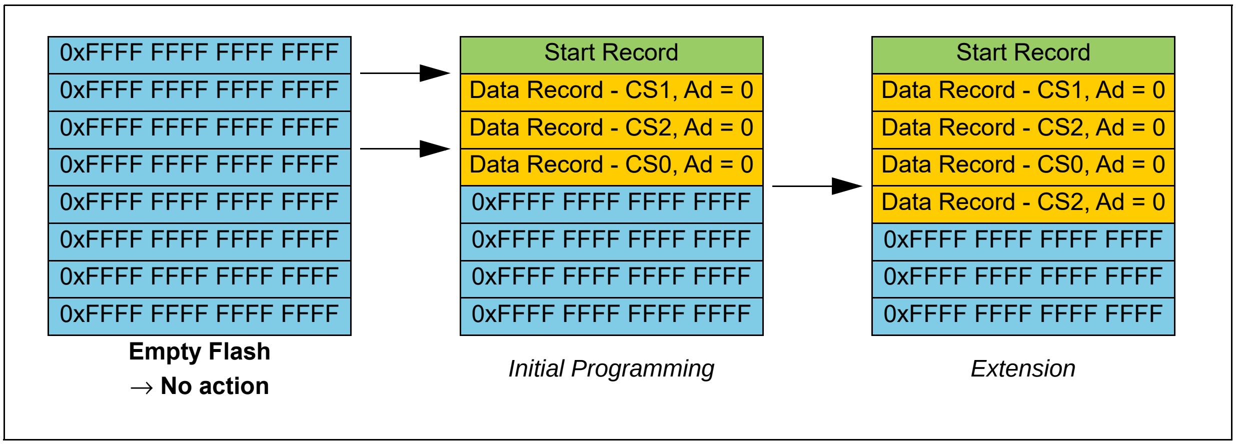
Table 106. Series of DCF records in UTEST Flash memory (continued)

ADDR offset	DATA			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8*n* + 0x0	Reserved			
8*n* + 0x4	Reserved			STOP=1
8(*n*+1) + 0x0	—			
8(*n*+1) + 0x4				

Caution: There must never be an unprogrammed record in the DCF data structure, as it is interpreted as a stop record and subsequent records are ignored.

Records programmed in several sessions is shown in *Figure 38*, appending new records to the end of the list each time.

Figure 38. Appending DCF records



It is possible for several DCF Records to write to the same DCF client. The later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

9.3.1 UTEST DCF records

UTEST DCF records are located in the UTEST Flash memory area (refer to UTEST Flash memory map in the Memory map chapter). Some UTEST DCF records are programmed at the factory and the user may add USER DCF records from the first unprogrammed location following the factory programmed UTEST DCF records.

When programming UTEST DCF records, the records must start at the first address in the UTEST Flash memory area and be continuous: one UTEST DCF record must immediately follow the previous UTEST DCF record. An unprogrammed location in the UTEST Flash memory area is interpreted by the SSCM as the end of the UTEST DCF records. When this happens, the SSCM passes control of the boot sequence to the Reset Generation Module.

9.4 DCF client table

A list of DCF clients is shown in *Table 107*. Various attributes of the DCF clients are listed in these tables:

- DCF Chip Select [14:0]: the CS field is a 15-bit field with each bit acting as a module select signal for the internal modules. Only one bit in this field should be set in a DCF record.
- DCF address [16:2]: the 15-bit ADDR field specifies an internal address for a DCF client. These addresses are only used by the SSCM when interpreting DCF records and writing data to specified modules.
- DCF Client Description: the name of the DCF clients.
- Reset value of DCF client: the reset value of DCF client before being written by the SSCM with the associated DCF record.
- IPS read: this column specifies whether or not a DCF client can be read by user software after the release of reset.
- DCF client special strategy: registers are designated as DCF clients if they need to be written with a DCF record. Other features that must be designated for DCF clients include different write strategies listed below. The following list defines the special strategies:
  - None: no special DCF strategy is used.
  - Write Once: a DCF client can only be written once. The DCF client ignores subsequent writes.
  - Triple Voted: DCF clients have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value. Triple voting allows for ‘bit-flip’ error tolerance without changing the DCF client output data.
  - Write 0 only: a bit in a DCF client can only be written from a logic 1 to a logic 0. An attempt to write a logic 1 is ignored.
  - Write 1 only: a bit in a DCF client can only be written from a logic 0 to a logic 1. An attempt to write a logic 0 is ignored.
- DCF order dependency in Flash memory
  - DCF order dependency in Flash memory: this column details whether or not there are special conditions on the location of a particular DCF record in Flash memory.

9.4.1 DCF client list

Table 107. DCF client list[1]

DCF CS [14:0]	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
(Binary)							
SSCM							
000_0000_0000_0010	0000000	00000101	BAF Configuration (Bypass mode) (see *Figure 39* and *Table 108*)	0xF7FFC002	yes	None	None
STCU - Support for 35 Memory Cuts (NMCUT)							
000_0000_0000_0100	0000000	00000010	STCU_SKC	0x00000000	yes	None	First record among STCU, Key1/Key2 DCF record sequence; Write Key2 every (1024-8) clock edges
000_0000_0000_0100	0000000	00000000	STCU_RUN	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000011	STCU_CFG	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000100	STCU_PLL_CFG	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000101	STCU_WDG	0x0000FFFF	yes	None	None
000_0000_0000_0100	0000000	00000111	STCU_CRCE	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00001010	STCU_ERR_FM	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00100001	STCU_MBUFM1[31:0][2]	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00100010	STCU_MBUFM2[31:0][2]	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000000	STCU_MB_CTRL_0	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000001	STCU_MB_CTRL_1	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000010	STCU_MB_CTRL_2	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000011	STCU_MB_CTRL_3	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000100	STCU_MB_CTRL_4	0x00000000	yes	None	None

Table 107. DCF client list(1) (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0000_0100	0000001	10000101	STCU_MB_CTRL_5	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000110	STCU_MB_CTRL_6	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000111	STCU_MB_CTRL_7	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001000	STCU_MB_CTRL_8	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001001	STCU_MB_CTRL_9	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001010	STCU_MB_CTRL_10	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001011	STCU_MB_CTRL_11	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001100	STCU_MB_CTRL_12	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001101	STCU_MB_CTRL_13	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001110	STCU_MB_CTRL_14	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001111	STCU_MB_CTRL_15	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010000	STCU_MB_CTRL_16	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010001	STCU_MB_CTRL_17	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010010	STCU_MB_CTRL_18	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010011	STCU_MB_CTRL_19	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010100	STCU_MB_CTRL_20	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010101	STCU_MB_CTRL_21	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010110	STCU_MB_CTRL_22	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010111	STCU_MB_CTRL_23	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011000	STCU_MB_CTRL_24	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011001	STCU_MB_CTRL_25	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011010	STCU_MB_CTRL_26	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011011	STCU_MB_CTRL_27	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011100	STCU_MB_CTRL_28	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011101	STCU_MB_CTRL_29	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011110	STCU_MB_CTRL_30	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011111	STCU_MB_CTRL_31	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100000	STCU_MB_CTRL_32	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100001	STCU_MB_CTRL_33	0x00000000	yes	None	None

Table 107. DCF client list[(1)] (continued)

DCF CS [14:0]	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
(Binary)							
000_0000_0000_0100	0000001	10100010	STCU_MB_CTRL_34	0x00000000	yes	None	None
PASS							
000_0000_0000_1000	0000000	00101000 -00101011	Reserved				
000_0000_0000_1000	0000000	00101100	Censorship	0x0	yes	None	None
000_0000_0000_1000	0000000	00101101	Password Slot Access	0x0	yes	wr_once	None
000_0000_0000_1000	0000000	00110000	Reserved				
000_0000_0000_1000	0000000	00110001-00110111	Reserved				
000_0000_0000_1000	0000000	00111000	Fuse Bypass Enable	0x1	yes	wr_once	None
000_0000_0000_1000	0000000	00111001-00111111	Reserved				
000_0000_0000_1000	0000000	01000000	LOCK0_PG0	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01000001	LOCK1_PG0	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01000010	LOCK2_PG0	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01000011	LOCK3_PG0	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01000100	LOCK0_PG1	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01000101	LOCK1_PG1	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01000110	LOCK2_PG1	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01000111	LOCK3_PG1	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001000	LOCK0_PG2	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001001	LOCK1_PG2	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001010	LOCK2_PG2	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001011	LOCK3_PG2	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001100	LOCK0_PG3	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001101	LOCK1_PG3	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001110	LOCK2_PG3	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001111	LOCK3_PG3	0xFFFFFFFF	yes	None	None
000_0000_0000_1000	0000000	01001100-11111111	Reserved				

Table 107. DCF client list[1] (continued)

DCF CS [14:0]	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
(Binary)							
Tamper Detect							
000_0000_0001_0000	0000000	00000000	Diary Base Address	0xFFFFFFFF	no	wr_once + wr0_only	None
000_0000_0001_0000	0000000	00000001	Tamper Region Override	0x00	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00000010	Software Tamper Region Override Disable	0x00	no	wr1_only	None
000_0000_0001_0000	0000000	00000011-00000111	Reserved				
000_0000_0001_0000	0000000	00001000	OTP_EN0	0x00000000	no	wr1_only	None
000_0000_0001_0000	0000000	00001001	OTP_EN1	0x00000000	no	wr1_only	None
000_0000_0001_0000	0000000	00001010	OTP_EN2	0x00000000	no	wr1_only	None
000_0000_0001_0000	0000000	00001011	OTP_EN3	0x00000000	no	wr1_only	None
000_0000_0001_0000	0000000	00001100-00010011	Reserved				
000_0000_0001_0000	0000000	00010100	TDR0_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00010101	TDR0_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00010110	TDR0_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00010111	TDR0_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011000	TDR1_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011001	TDR1_LOCK1	0x00000000	no	wr_once + wr1_only	None

Table 107. DCF client list[(1)] (continued)

DCF CS [14:0]	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
(Binary)							
000_0000_0001_0000	0000000	00011010	TDR1_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011011	TDR1_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011100	TDR2_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011101	TDR2_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011110	TDR2_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011111	TDR2_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100000	TDR3_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100001	TDR3_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100010	TDR3_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100011	TDR3_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100100	TDR4_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100101	TDR4_LOCK1	0x00000000	no	wr_once + wr1_only	None

Table 107. DCF client list[1] (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0001_0000	0000000	00100110	TDR4_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100111	TDR4_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101000	TDR5_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101001	TDR5_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101010	TDR5_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101011	TDR5_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101100-11111111	Reserved				
HSM							
000_0000_0010_0000	0000000	00001000 -00001111	Reserved				
000_0000_0010_0000	0000000	00010000	HSM Enable and Configuration	0x00000000	no	wr_once	None
000_0000_0010_0000	0000000	00010001	HSM Software Inhibit	0x00000000	no	wr_once	None
000_0000_0010_0000	0000000	00010010	Flash Test Mode Protection	0x00000002	no	Triple Voted + wr_once	None
000_0000_0010_0000	0000000	00010011-00011111	Reserved				
000_0000_0010_0000	0000000	00100000	HSM Exclusive Flags	0x00000000	no	wr_once	None
000_0000_0010_0000	0000000	00100001 -00111111	Reserved				
000_0000_0010_0000	0000000	01000000	HSM Alternate Prog Interface Flags	0x00000000	no	wr_once	None

Table 107. DCF client list[1] (continued)

DCF CS [14:0]	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
(Binary)							
000_0000_0010_0000	0000000	01000001 -11111111	Reserved				
000_0000_0010_0000	0000001 -1111111	00000000 -11111111	Reserved				
MISCELLANEOUS							
000_0000_1000_0000	0000000	00000000	UTEST Miscellaneous (See *Figure 40* and *Table 109* for bit position details.)	0x00049E01	no	Triple Voted	None
000_0000_1000_0000	0000000	00000001	UTEST Miscellaneous_2 (See *Figure 41* and *Table 110* for bit position details.)	0x00000000	no	Triple Voted + wr_once	None
000_0000_1000_0000	0000000	00000010 -00000110	Reserved				
000_0000_1000_0000	0000000	00000111	JTAG Pin Configuration	0x00000001	no	None	None
000_0000_1000_0000	0000000	00001000 -01000111	Reserved				
000_0000_1000_0000	0000000	01001000	ESR0_CONFIG (See ESR0_CFG register section in PMC Digital Interface chapter)	0x00000000	yes	Triple Voted + chk_wr2	None
000_0000_1000_0000	0000000	01001001	PMC REE_BUS	0x00000000	no	Triple Voted	None
000_0000_1000_0000	0000000	01001010 -11111111	Reserved				
000_0000_1000_0000	0000001 -1111111	00000000 -11111111	Reserved				
SOFT DCF CLIENTS (BAF)							
100_0000_0000_0000	0000000	00000000	Security Watchdog Control Register	NA			Refer to *Chapter 52: Boot Assist Flash (BAF)*

Table 107. DCF client list[1] (continued)

DCF CS [14:0]	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
(Binary)							
100_0000_0000_0000	0000000	00000100 -11111111	Reserved				
100_0000_0000_0000	0000001	00000000 -11111111	Reserved				
100_0000_0000_0000	0000010	00000000	Address	NA			None
100_0000_0000_0000	0000010	00000001	32-bit Data	NA			None
100_0000_0000_0000	0000010	00000010	16-bit Data	NA			None
100_0000_0000_0000	0000010	00000011	8-bit Data	NA			None
100_0000_0000_0000	0000010	00000100 -00001111	Reserved				
100_0000_0000_0000	0000010	00010000	Callback address	NA			None
100_0000_0000_0000	0000010	00010010 - 00010100	Reserved				
100_0000_0000_0000	0000011 -1111111	00000000 -11111111	Reserved				

1. DCF programming should start from address 0x00400208.
2. If this register is programmed for unrecoverable fault generation and LBIST/MBIST failure occurs, in offline self test, it will cause the device to get stuck in reset mode

9.4.2 BAF configuration DCF register

Offset: 00000000 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BAF_ADDRESS															
W																
Reset	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BAF_ADDRESS														Reserved	BAF_BYPASS
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 39. BAF configuration DCF Client

Table 108. BAF configuration DCF Client field descriptions

Field	Description
0:29 BAF_ADDRESS	BAF start address
31 BAF_BYPASS	BAF bypass 0 BAF is executed 1 BAF is bypassed

9.4.3 Miscellaneous DCF registers

Offset: 00000000 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	Reserved													XOSC_IBIAS_SEL[2:0]		

Reset Value is chip-specific; refer to *Table 107: DCF client list*.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	Reserved	XOSC_ALC_DIS	Reserved	XOSC_FREQ_SEL[2:0]			XOSC_AUTOSWITCHB	Reserved	XOSC_EN	XOSC_EXT_CLOAD	XOSC_LOAD_CAP_SEL[4:0]					Reserved

Reset Value is chip-specific; refer to *Table 107: DCF client list*.

Figure 40. UTEST Miscellaneous DCF Client

Table 109. UTEST Miscellaneous DCF Client field descriptions

Field	Description
13:15 XOSC_IBIAS_SEL[2:0]	Allows to change oscillation margin (default “100”) - User to keep default value of “100”
17 XOSC_ALC_DIS	Active high to disable Automatic Level Controller
19:21 XOSC_FREQ_SEL[2:0]	This will select the different frequency settings of XOSC. 000 4 MHz – 8 MHz 001 5 MHz – 10 MHz 010 10 MHz – 15 MHz 011 15 MHz – 20 MHz 100 20 MHz – 25 MHz 101 25 MHz – 30 MHz 110 30 MHz – 35 MHz 111 35 MHz – 40 MHz
22 XOSC_AUTOSWITCHB	Allows to enable or disable the autoswitch 0 Reserved 1 Autoswitch mode disabled

Table 109. UTEST Miscellaneous DCF Client field descriptions (continued)

Field	Description
24 XOSC_EN	Enable XOSC 0 XOSC disabled 1 XOSC enabled
25 XOSC_EXT_CLOAD	XOSC EXT CLOAD 0 Selects XOSC internal cap (recommended for 20 MHz – 40 MHz operation) 1 Selects XOSC external cap (recommended for 4 MHz – 20 MHz operation) **Note:** It is always recommended to verify the required size of the capacitances with the crystal supplier.
26:30 XOSC_LOAD_CAP_SEL[0:4]	XOSC LOAD CAP SEL [0:4] Five trim bits program the load capacitance. The formula to calculate the capacitance offered is $C_{tot} = C_{fix}+n.C_u$ where: Fixed Capacitance C_{fix} = 11.0pF (variation ±15%) Unit capacitance C_u = 0.49pF (variation ±15%) n = load_cap_sel[4] * 2^4 + load_cap_sel[3] * 2^3 + load_cap_sel[2] * 2^2 + load_cap_sel[1] * 2^1 + load_cap_sel[0] * 2^0

Offset: 00000001 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	Reserved													CFLA_ON_BYPASS_MC_ME	CFLA_ON_AFTER_STANDBY	BAF_EXEC_DURING_FA
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41. UTEST2 Miscellaneous DCF Client

Table 110. UTEST2 Miscellaneous DCF Client field descriptions

Field	Description
29 CFLA_ON_BYPASS_MC_ME	1 Flash memory cannot be switched-off by MC_ME.CFLAON bits in RUNx modes 0 Flash memory can be switched-off by MC_ME.CFLAON bits in RUNx modes
30 CFLA_ON_AFTER_STANDBY	1 Flash memory will be turn-on after the STDBY exit 0 Flash memory can be switched (on/off) by MC_ME.CFLAON bits
31 BAF_EXEC_DURING_FA	1 BAF does not stop into an endless loop when LifeCycle = FA 0 BAF stops into an endless loop when LifeCycle = FA

Offset: 00000111 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved													JTAG_PIN_CFG_HYSE	JTAG_PIN_CFG_ILS[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 42. JTAG Pin Configuration

Table 111. JTAG Pin Configuration field descriptions

Field	Description
29 JTAG_PIN_CFG_HYSE	JTAG pins hysteresis 0 JTAG pins hysteresis not enabled. 1 JTAG pins hysteresis enabled. Default: JTAG pins hysteresis not enabled.
30:31 JTAG_PIN_CFG_ILS	JTAG pins mode 00 JTAG pins configured for AUTO mode. 01 JTAG pins configured for TTL mode. 10 JTAG pins configured for TTL mode. 11 JTAG pins configured for CMOS mode. Default: JTAG pins configured for TTL mode.

Offset: 01001001 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															LVD14_AS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVD14_IF	0	LVD14_IM	HVD13_IF	0	0	0	LVD11_AS	LVD11_IF	0	LVD11_FL	LVD11_C	HVD6_C	LVD3_SB	LVD3F_L	LVD3_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43. PMC_REE_BUS

Table 112. PMC_REE_BUS field descriptions

Field	Description
15 LVD14_AS	This bit defines whether an LVD assertion on the first supply of the ADC generates a system reset. 0 Disabled. LVD assertion on the first supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the first supply of the ADC causes system reset.
16 LVD14_IF	This bit defines whether an LVD assertion on the first supply of the IO Ethernet generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO Ethernet does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO Ethernet causes system reset.
18 LVD14_IM	This bit defines whether an LVD assertion on the first supply of the IO mains generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO mains does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO mains causes system reset.
19 HVD13_IF	This bit defines whether an HVD assertion on the first supply of the IO Ethernet generates a system reset. 0 Disabled. HVD assertion on the first supply of the IO Ethernet does not cause system reset. 1 Enabled. HVD assertion on the first supply of the IO Ethernet causes system reset.
23 LVD11_AS	This bit defines whether an LVD assertion on the first supply of the SARADC generates a system reset. 0 Disabled. LVD assertion on the first supply of the SARADC does not cause system reset. 1 Enabled. LVD assertion on the first supply of the SARADC causes system reset.

Table 112. PMC_REE_BUS field descriptions (continued)

Field	Description
24 LVD11_IF	This bit defines whether an LVD assertion on the first supply of the IO Ethernet generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO Ethernet does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO Ethernet causes system reset.
26 LVD11_FL	This bit defines whether an LVD assertion on the first supply of the Flash generates a system reset. 0 Disabled. LVD assertion on the first supply of the Flash does not cause system reset. 1 Enabled. LVD assertion on the first supply of the Flash causes system reset.
27 LVD11_C	This bit defines whether an LVD assertion on the first supply of the core generates a system reset. 0 Disabled. LVD assertion on the first supply of the core does not cause system reset. 1 Enabled. LVD assertion on the first supply of the core causes system reset.
28 HVD6_C	This bit defines whether an HVD assertion on the first supply of the core generates a system reset. 0 Disabled. HVD assertion on the first supply of the core does not cause system reset. 1 Enabled. HVD assertion on the first supply of the core causes system reset.
29 LVD3_SB	This bit defines whether an LVD assertion on the first supply of the IO ring generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO ring does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO ring causes system reset.
30 LVD3_FL	This bit defines whether an LVD assertion on the first supply of the Flash generates a system reset. 0 Disabled. LVD assertion on the first supply of the Flash does not cause system reset. 1 Enabled. LVD assertion on the first supply of the Flash causes system reset.
31 LVD3_C	This bit defines whether an LVD assertion on the first supply of the core generates a system reset. 0 Disabled. LVD assertion on the first supply of the core does not cause system reset. 1 Enabled. LVD assertion on the first supply of the core causes system reset.

Offset: 000000000000000 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved														eFE	FTMPL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 44. Flash test mode protection

Table 113. Flash test mode protection field descriptions

Field	Description
30 eFE	0 Efuse system is disabled. 1 Efuse system is enabled. Default: Efuse system is enabled.
31 FTMPL	0 FTMP is not locked. 1 FTMP is locked. Default: FTMP is not locked.

10 Power management

10.1 Overview

SPC584Bx microcontrollers include a robust power management infrastructure that enables applications to select among various operational and low-power modes and to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can leave reset. This chapter gives an overview of the built-in power management features.

The power management infrastructure comprises the following modules:

- Power Control Unit (MC_PCU)
- Clock Generation Module (MC_CGM)
- Mode Entry Module (MC_ME)
- Power Management Controller (PMC)
- Power Management Controller Digital Interface (PMC_DIG)
- Smart Standby Wake-up Unit (SSWU)

This chapter describes these modules in brief. For complete details, refer to the respective module chapters.

The MC_PCU provides a status register and a submodule, the PMC_dig (Power Management Control Digital Interface), provides an interface to configure and control internal voltages.

The MC_CGM generates reference clocks for all the chip blocks. It works in conjunction with internal clock gating logic to implement low power modes.

The MC_ME module controls chip operational modes and mode transition sequences. It also contains configuration, control and status registers accessible for the application.

The PMC module contains all the internal voltage regulator and monitors of the device: refer to *Section 10.1.3: Power management controller overview*.

The PMC_DIG module is the digital interface of the PMC, and contains all the registers to control the PMC during reset and power mode changes.

The SSWU Unit provides services to schedule sequence of tasks executed without CPU intervention. This is aimed at reducing further power consumption during standby.

The SPC584Bx has two types of modes, SYSTEM modes and USER modes.

SYSTEM modes:

- SYSTEM modes are entered automatically on system events or on application request.
- The SYSTEM modes are: RESET, SAFE, TEST and DRUN.

USER modes:

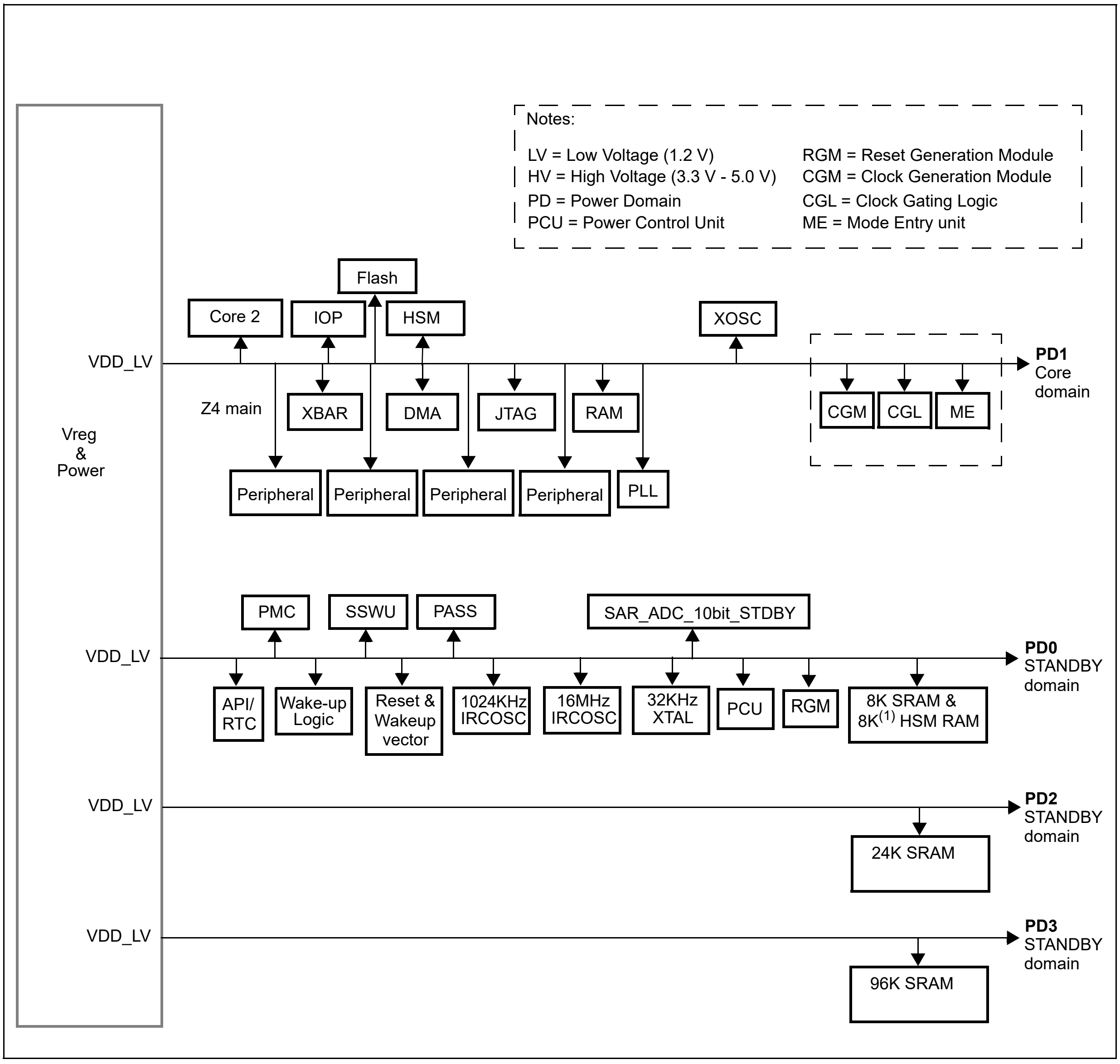
- USER modes control performance and consumption during normal application operation. The modes are organized according to the performance provided
- The USER modes are RUN0-3, HALT, STOP and STANDBY.
- Each mode is entered by writing a dedicated register protected by a key, to avoid unwanted transition

Please refer to the Magic Carpet Mode Entry Module chapter for modes details.

10.1.1 Power management framework

The power management framework supports a variety of configuration options. Refer to *Figure 45* for more details. SPC584Bx implements a unique core voltage power domain. There are no separate functional power domains.

Figure 45. Device power management framework



1. Enabled during stand-by depending on a DCF bit

10.1.2 Power management supply description

Table 114 provides an overview of the PMC supply signals.

Table 114. Device power management controller external signals

Name	Type	Description
VDD_HV_ADR_S	Reference	Voltage reference of ADC SAR module
VDD_HV_ADV	Supply	High voltage supply for the ADC modules
VDD_HV_FLA	Supply	Supply pin for Flash
VDD_HV_IO_MAIN	Supply	High voltage power supply for I/O
VDD_HV_IO_ETH	Supply	Ethernet I/O supply
VDD_LV	Supply	Low voltage power supply for the core area generated by internal regulator
VSS	Ground	Ground supply for the device / I/O. This is covering both VSS_LV and VSS_HV in case of exposed pad device.
VSS_HV_ADR_S	Reference	Ground reference of ADC SAR module
VSS_HV_ADV	Supply	Ground supply for the ADC modules

Caution: There are constraints on some input voltages relative to other input voltages. Refer to the device data sheet for detailed information.

10.1.3 Power management controller overview

The power management controller unit (PMC) contains all the device internal voltage regulators and all the voltage monitors. The supported regulators are:

- Internal regulator
  - An internal high power regulator (HPREG) using an external NPN bipolar as the ballast device (Refer to the SPC584Bx datasheet for the bipolar part number).
  - An internal digital regulator (DREG).
- Low-power regulator
  - When the device is in STANDBY, the supply is provided by the internal LPREG and the HPREG/DREG is off, with a reduced maximum driving capability.
- Auxiliary and clamp regulators
  - For fast current transients, auxiliary and clamp regulators are used in both HPREG and DREG modes to improve the load regulation by compensating voltage overshoots and under-shoots.

Please refer to the data sheet for the electrical characteristics of the regulators and the monitors and the list of external components required. The list and functionality of the monitors are described further in this chapter.

Figure 46 describes the top level block diagram of power management unit.

Figure 46. Power management controller block diagram overview

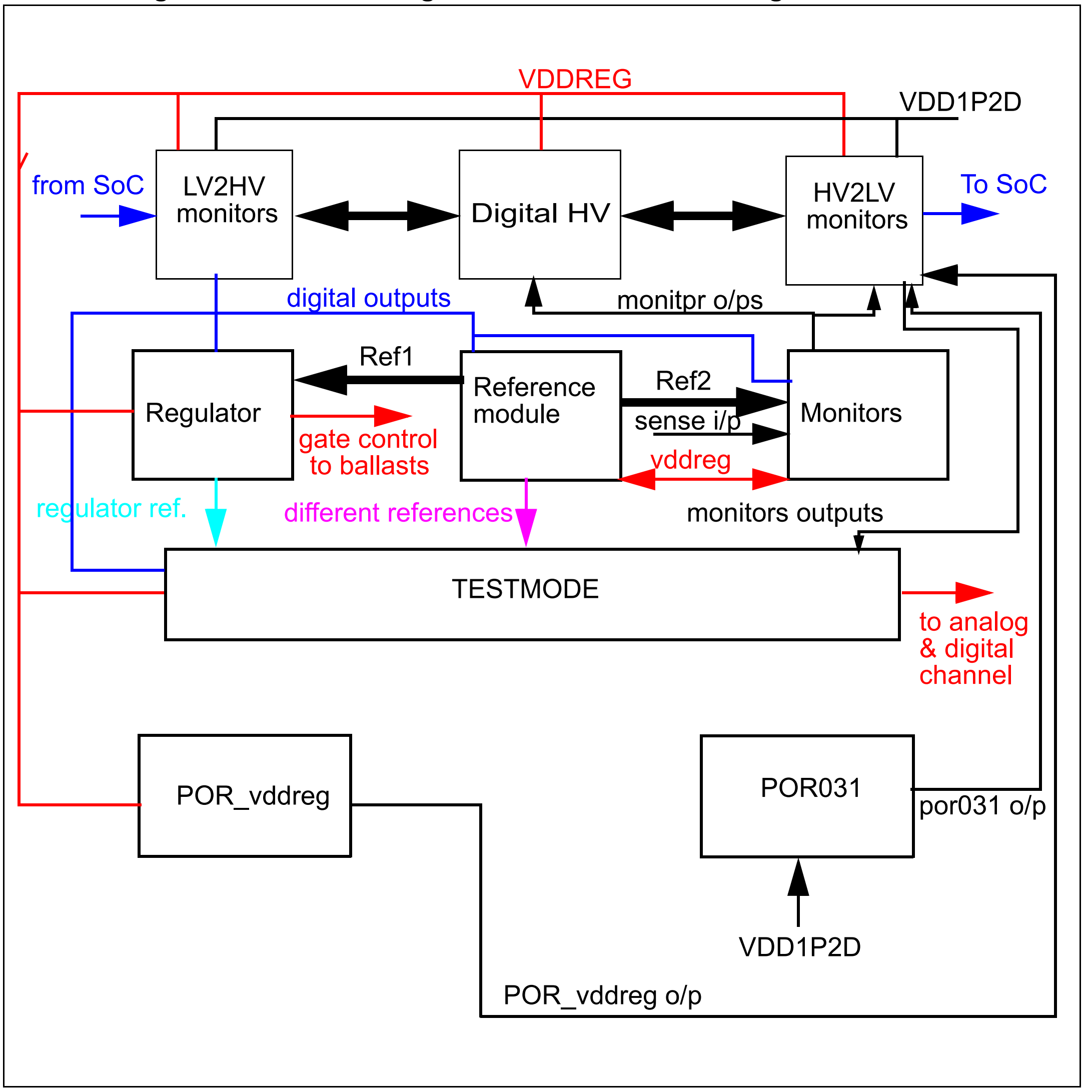
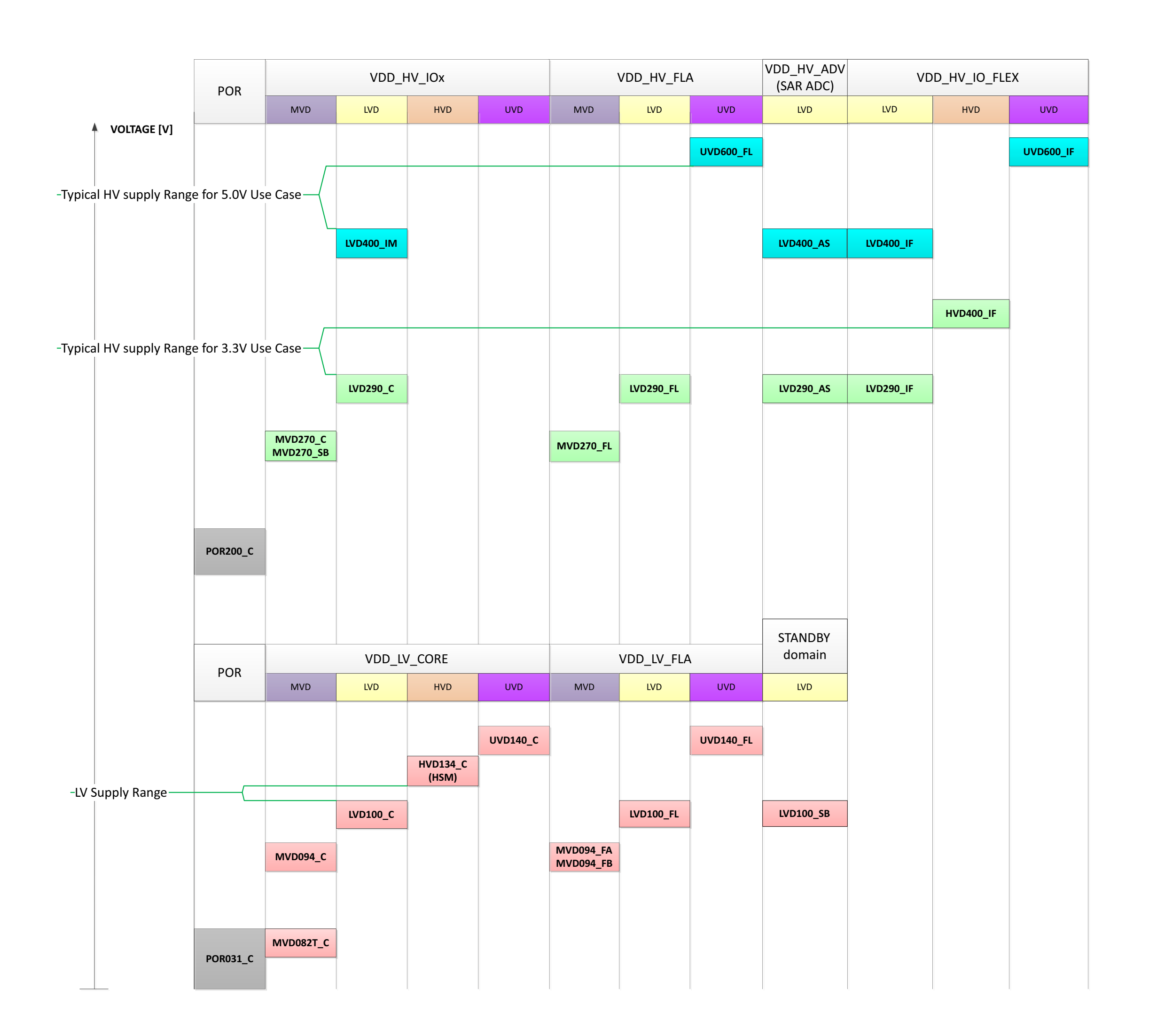


Figure 47. Voltage Monitor Coverage(1)(2)



1. Refer to *Table 115* for VDs descriptions
2. Voltage levels for all VDs are given in the data sheet

VDD_HV_IO is redundant, ensuring that single pin failure does not prevent device from functioning correctly. Full specification may not be guaranteed.

VDD_LV is redundant, ensuring that single pin failure does not prevent the device from functioning correctly.

10.1.3.1 1.2 V safety/spike regulator specification

SPC584Bx implements a 1.2 V safety/spike regulator. This regulator can be used to handle fast application current variation that cannot be fully handled by an external regulator.

10.2 Low power mode support

The SPC584Bx microcontroller supports low power modes.

Dedicated logic prevents uncontrolled activation of low-power mode. This is implemented using protected sequence from ME.

Inadvertent entry into low power mode is prevented by triple voting of the control flops, both SW readable and otherwise.

10.2.1 Low power mode (HALT/STOP/STANDBY)

Low power mode may be used to control the ECU during the IDLE phase, as in a partial networking system or when the key has been turned off and periodic tasks, like contact monitoring, must be maintained.

Note: STANDBY mode behaves very differently from HALT/STOP modes. For more details, refer to Section 55.4.4.1: STANDBY Mode transition and Section 56.4.2.8: STANDBY0 Mode.

The device implements the equivalent of an idle or power-down mode through STANDBY, STOP and HALT mode configuration.

The entry into a low power mode is requested by the application code while the exit is triggered by a hardware event (internal, for instance RTC/API, or external, for instance Wake-up signal) or by the SSWU which detected a wake-up condition.

A combination of mode control and clock control (clock source selection but also peripheral clock gating) allows the user to configure the low power modes according to his application needs.

In STOP/HALT mode, peripheral modules can be enabled/disabled using the control described in the ME section. For example, it is possible to keep two CAN modules, eMIOS module and a few requested pins active. For alternate configurations, other modules can be enabled/disabled using the control described in the ME section. External interrupt request channels and external wake-up channels can be made active to wake-up the device in alternative to peripherals interrupts.

All pins associated to the active modules are functional.

Control of the pins is defined by user during run time. A wake-up line is associated to each selectable module.

One of the available modules can be configured to perform the wake-up task. For example, with all but one CAN modules on the chip set completely to power down mode and only one is configured for wake-up message receive (any of the modules can be configured to perform this task).

All CAN modules can be independently configured to receive messages and respond to wake-up messages.

Basically the low power modes allow to control the clock and the state of all peripherals (analog and digital) but also select the mode of the flash module as well as select and enable/disable clock sources like PLL or oscillator.

10.2.1.1 ECU in SLEEP mode versus Contact Monitoring

Most Body ECU are requested to support a SLEEP mode which is activated while the car is stopped and switched off.

The activity to be maintained during the SLEEP mode depends on the application, while a BCM has to ensure the periodic contact monitoring task, a simple Junction Box may just monitor the battery voltage.

So in most cases a Body ECU remains supplied and active in SLEEP mode, therefore SPC584Bx offers a versatile module, called Smart Standby Wake-up Unit, to handle in STANDBY mode, though without CPU intervention, the periodic event monitoring tasks.

More details about those advanced low power modes are described in the SSWU and ME specific chapters.

10.3 Flash power requirements

During start-up operation, the flash module operates from an untrimmed regulator, which has a slightly larger variation than the ideal case. During this start-up phase, the flash can still be read, but it must be read at a slower rate. While operating in this slower read mode, the flash can operate at a much reduced voltage.

Following completion of the start-up phase the regulator is in its trimmed condition, and subsequently the flash module reference current has also been trimmed. Therefore when entering normal operation the flash can be read in the normal access manner and support the full flash specification.

10.4 Device trimming

During the initialization phase the device defaults to a pre-determined state for each of the LVDs, HVDs and the internal regulators. As the flash becomes available the differential read process allows the trimmed data to be available for trimming the internal LVDs, HVDs and regulators.

Please refer to *Section 10.6* for further details.

10.5 Supply monitoring (POR and LVDs)

The function of the POR and LVD circuits is to hold the device in reset regardless of how slow the supply voltage rise is, until the point at which the POR and LVDs are released.

The POR and LVD circuits function correctly even if the input voltage is non monotonic.

10.5.1 Power-on reset (POR)

The PMC implements two internal power-on reset circuits:

- POR031_C monitors the voltage on the 1.2 V input supply. It is monitoring the VDD_LV_CORE pin. The POR031_C asserts a reset when the input supply is below defined values.
- POR200_C monitors the voltage on the 5.0 V - 3.3 V input supply. It is monitoring the VDD_HV_IO_MAIN pin. The POR trip point is high enough to make sure all the LVD circuits are functional.

Refer to *Chapter 8: Reset and Boot* for PORST/ESR0 pin functionality.

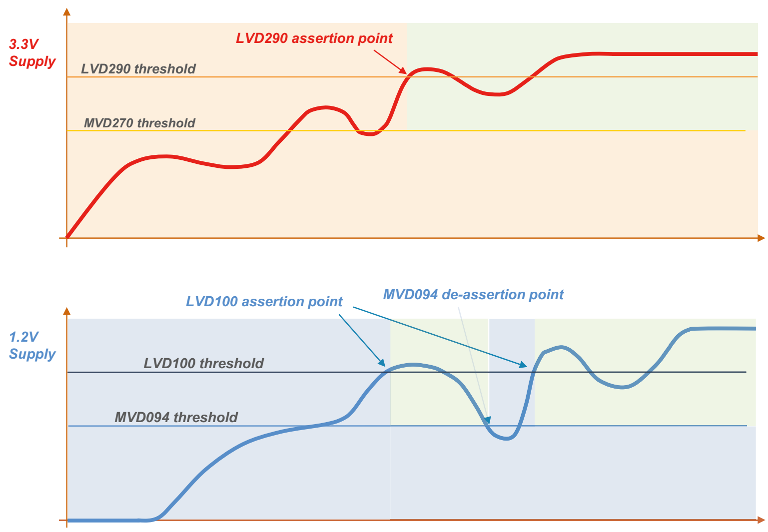
10.5.2 Behavior of device LVD / HVD

Although there is an option to disable the LVD and HVD following reset (refer to *Section 10.5.3*), they are capable of being used in a ‘monitor’ only mode and also capable of generating a safe / interrupt event. *Figure 48* illustrates the behavior of these internal LVD /

HVD circuits during the power on phase, through device initialization and subsequently in ‘normal’ run / operating mode.

The LVDs/HVDs can also be configured after device initialization preventing reset to happen when supply crosses the LVD threshold, effectively providing a higher voltage operating range. It is the responsibility of the application to ensure that the device remains in the functional range.

Figure 48. Internal LVD and HVD in configuration example during power-up



10.5.3 Voltage detections (MVDs, LVDs, HVDs, UVDs)

The internal LVD circuits, monitor when the voltage on the corresponding supply is below the defined values and either assert a reset or an interrupt. The LVDs also support hysteresis in the falling and rising trip points.

10.5.3.1 LVD and HVD implementation

- All LVDs and HVDs are capable of generating either an RGM and/or FCCU event and/or interrupt event.
- All LVDs and HVD configured for reset generation can cause functional or destructive reset. MC_RGM PHASE0 is not exited until all destructive reset conditions are cleared.
- The appropriate bits in the PMC registers are set by LVD and HVD events. Please refer to Power management controler digital interface chapter (PMC_Dig).
- The LVDs and HVDs control is protected by the register protection scheme. Therefore it is configurable as long as the scheme is followed. Refer to *Chapter 77: Register Protection (REG_PROT)*.
- SPC584Bx implements REE (Reset event enable) DCF records to allow to associate “configurable” LVDs/HVDs to a RESET event. This is a write mechanism managed by SSCM during device initialization. When RESET event is selected through REE DCF record, it cannot be disabled by software programming of PMC REE register. When RESET event is not selected through REE DCF record, LVDs/HVDs trigger event reaction depends on the programming of the PMC REE/RES/IE/FEE registers. Refer to *Chapter 9: Device Configuration Format (DCF) Records*.
- When the LVD or the HVD is enabled for destructive reset generation and a subsequent trigger event is detected, the external PORST pin is driven low.

Note: The full list of voltage monitors is provided in Table 115. The functionality and configurability are reported in Figure 49 and Table 116.

Note: MVDs/UVDs are not configurable.

Table 115. Voltage monitors description

Monitor name	Voltage reference number	Description	Monitor sensing
POWER MANAGEMENT UNIT VOLTAGE MONITORS: LOW VOLTAGE SUPPLY			
POR031_C	—	Low voltage supply power-on reset voltage monitor	
MVD082T_C	VD1	LV supply core minimum voltage detector	VDD_LV_CORE
MVD094_C	VD2	LV supply core minimum voltage detector	VDD_LV_CORE
MVD094_FA	VD2	LV supply flash minimum voltage detector	VDD_LV_FLA
MVD094_FB	VD2	LV supply flash minimum voltage monitor	VDD_LV_FLA
LVD100_C	VD3	LV supply core low voltage detector	VDD_LV_CORE
LVD100_FL	VD3	LV supply flash low voltage detector	VDD_LV_FLA
LVD100_SB	VD3	LV supply STANDBY low voltage detector	VDD_LV_STBY
HVD134_C	VD6	LV supply core high voltage detector	VDD_LV_CORE
UVD140_C	VD7	LV supply core upper voltage detector	VDD_LV_CORE
UVD140_FL	VD7	LV supply flash upper voltage detector	VDD_LV_FLA

Table 115. Voltage monitors description (continued)

Monitor name	Voltage reference number	Description	Monitor sensing
POWER MANAGEMENT UNIT VOLTAGE MONITORS: HIGH VOLTAGE SUPPLY			
POR200_C	—	High voltage supply power-on reset voltage monitor	
MVD270_C	VD10	HV supply core minimum voltage monitor	VDD_HV_IOx
MVD270_FL	VD10	HV supply flash minimum voltage monitor	VDD_HV_FLA
MVD270_SB	VD10	HV supply STANDBY minimum voltage monitor	VDD_HV_IOx (in Standby)
LVD290_C	VD11	HV supply core low voltage monitor	VDD_HV_IOx
LVD290_FL	VD11	HV supply flash low voltage monitor	VDD_HV_FLA
LVD290_AS	VD11	HV supply ADC low voltage monitor	VDD_HV_ADV (SAR ADC)
LVD290_IF	VD11	HV supply I/O low voltage monitor	VDD_HV_IO_ETH
HVD400_IF	VD13	HV supply I/O high voltage monitor	VDD_HV_IO_ETH
LVD400_IM	VD14	HV supply core low voltage monitor	VDD_HV_IOx
LVD400_AS	VD14	HV supply ADC low voltage detector	VDD_HV_ADV (SAR ADC)
LVD400_IF	VD14	HV supply I/O main low voltage detector	VDD_HV_IO_ETH
UVD600_FL	VD15	HV supply flash upper voltage monitor	VDD_HV_IO_FLASH
UVD600_IF	VD15	HV supply I/O upper voltage monitor	VDD_HV_IO_ETH

Figure 49 provides the functionality of monitors depending on configuration.

Figure 49. Voltage monitors functionality

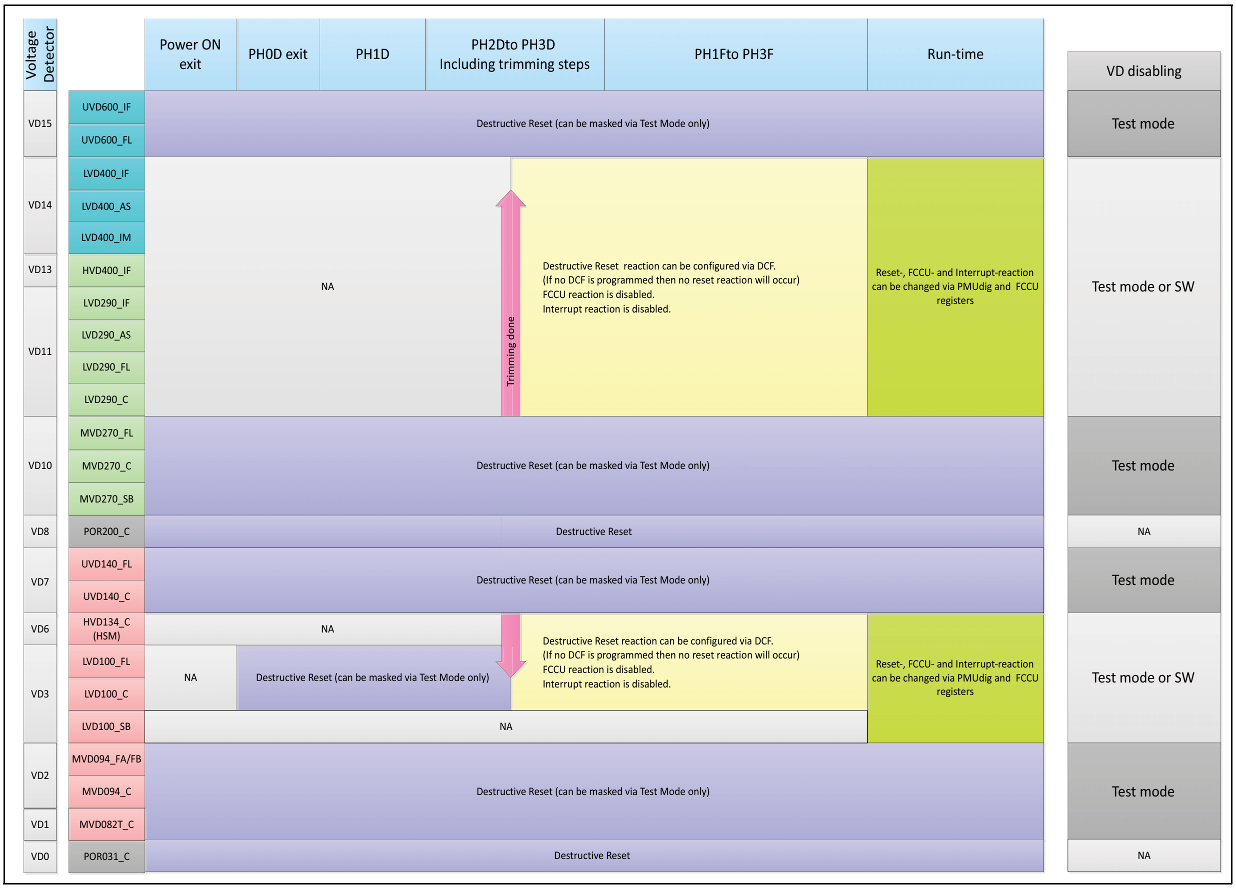


Table 116 provides the monitor status depending on configuration.

Table 116. Voltage monitors configurability

Monitor type	Reset Event Enable(1)	Reset Event Select(1)	Event Pending Register(1)	Interrupt Enable(1)	FCCU Event Enable(1)	Reset Event Enable DCF(1)
MVDs	No	No	No	No	No	No
LVDs	Yes	Yes	Yes	Yes	Yes	Yes
HVDs	Yes	Yes	Yes	Yes	Yes	Yes
UVDs	No	No	No	No	No	No

1. Refer to the Power management controller digital interface (PMC_Dig) chapter.

10.6 Power sequence

The following sections describe the power sequence and the relation between the different supplies during power-up and power-down.

The device is considered to be in a power sequence (or POWERUP state) when it is either not supplied or partially supplied. An internal power-on signal is used to identify POWERUP state. This signal is released high on exit of power sequence. The power-on signal is a combination of the LVDs monitoring the following supplies:

- VDD_LV
- VDD_HV_IO
- VDD_HV_FLA

The actual threshold use for each LVD is dependent on the configuration of the device. This is configurable by hardware (flash option bits content) or by software (LVD event configuration through register interface). Once power-on signal has been asserted, device configuration is reset to default power-up configuration.

10.6.1 Power-up sequence

In this section it is assumed that all supplies are low when entering the power-up sequence. Brown-out and power down sequences are managed in specific sections.

10.6.1.1 POWERUP state

At the beginning of power-up, the internal power-on signal remains low due to internal parasitics diodes. As soon as minimum threshold is reached on VDD_LV and VDD_HV supply, the internal power-on signal is forced low until power-up LVDs reach upper not trimmed threshold (refer to SPC584Bx datasheet).

The different supplies can rise independently as long as the constraints described in the SPC584Bx datasheet are met.

During power-up, all functional terminals are maintained into a known state as described in the SPC584Bx datasheet.

10.6.1.2 POWERUP exit

The POWERUP state can be exited when both VDD_LV and VDD_HV are above the internal LVD thresholds.

10.6.1.2.1 VDD_LV conditions and LVD trimming/enabling sequence

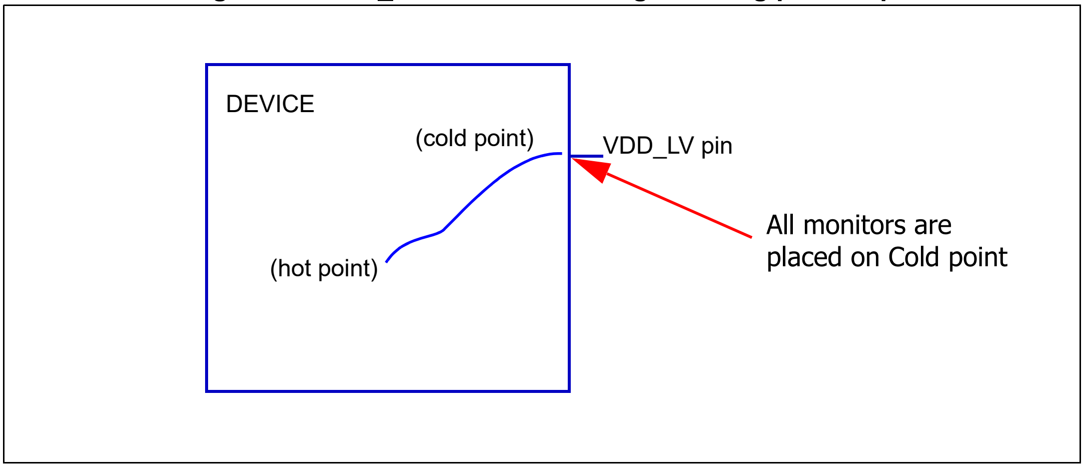
During POWERUP, only the following low voltage VDs are enabled:

- MVD082T_C , MVD094_C , MVD094_FA, MVD094_FB , UVD140_C , UDV140_FL

The VDD_LV conditions to exit POWERUP are the following:

- MVD094_C, MVD094_FA, MVD094_FB upper threshold is crossed

Figure 50. VDD_LV monitored voltages during power-up



After the POWERUP exit conditions (low voltage, high voltage conditions) have been verified, the internal power-on signal is released to all analog modules.

The internal RC oscillator module (IRCOSC) starts initialization and provides clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles and LVD100_C is masked. Only MVD094_C remains active.

MVD094_C and LVD100_C share the same reference. This difference provides margin with respect to maximum internal resistive drop and hysteresis addressing external supply.

The device proceeds through the reset sequence through RGM phase PHASE0, PHASE1[DEST], PHASE2[DEST] and PHASE3[DEST].

Voltage detector (LVD/HVD) modules are trimmed at the beginning of PHASE3[DEST]. Trimming of LVDs/HVDs are stored as internal DCF records. They are read by SSCM at low voltage, using the differential flash read accesses and applied to each LVDs/HVDs module. After analog delay ($t_{LVDTRIM}$) has elapsed, the PMC acknowledges that all LVDs/HVDs have been trimmed and supply is above threshold, the SSCM proceeds with the reset sequence, eventually running full speed accesses to the flash to read the option bits required to complete device configuration.

The configurable LVD/HVD modules can optionally be unmasked at the end of PHASE3[DEST]. Mask information is read as DCF record from the flash UTEST option bits.

When LVD100_C and LVD290_C are masked by the application using the flash user option bits, the device should rely on PORST signal to detect a voltage failure during power-up. The device must wait for PORST to be released high before proceeding with the power-up sequence.This may increase the amount of time necessary to complete the reset sequence.

Figure 51 provides an example of VDD_LV and VDD_HV power-up sequence before and after the monitor trimming.

Figure 52 provides an example of VDD_LV power-up sequence before and after the monitor trimming including UVD and HVD monitors.

Figure 51. Example of VDD_LV power-up sequencing

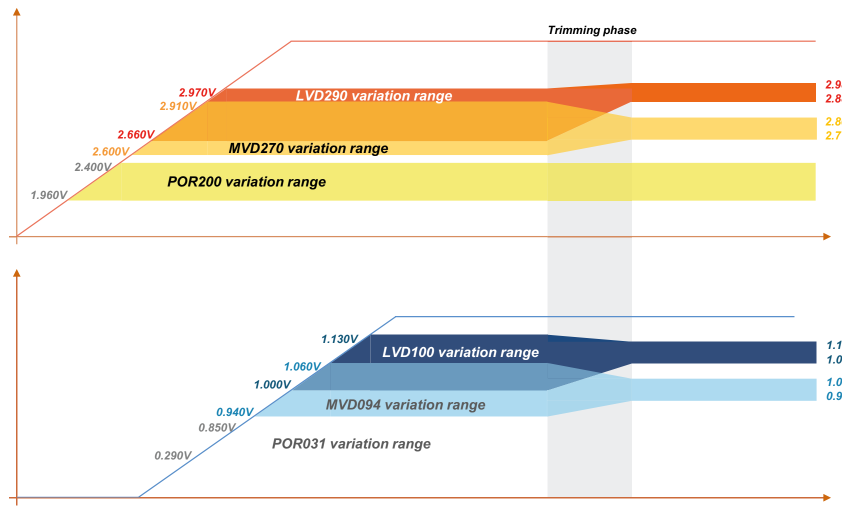
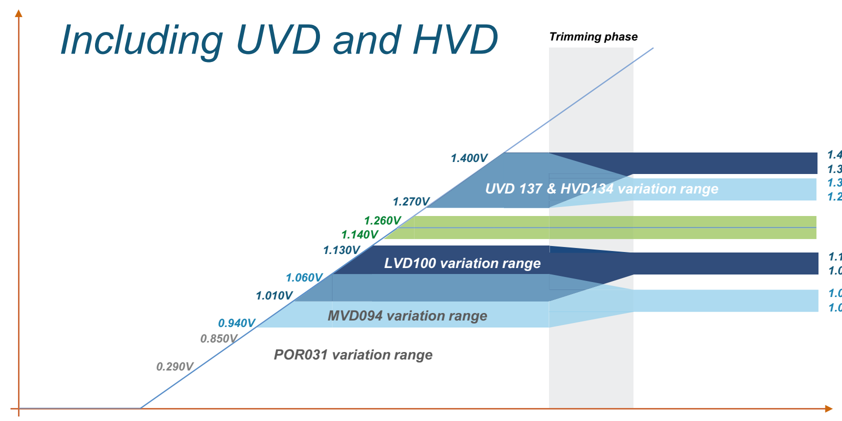


Figure 52. Threshold variation during power-up sequence on VDD_LV supply



10.6.1.2.2 VDD_HV conditions and LVD trimming/enabling sequence

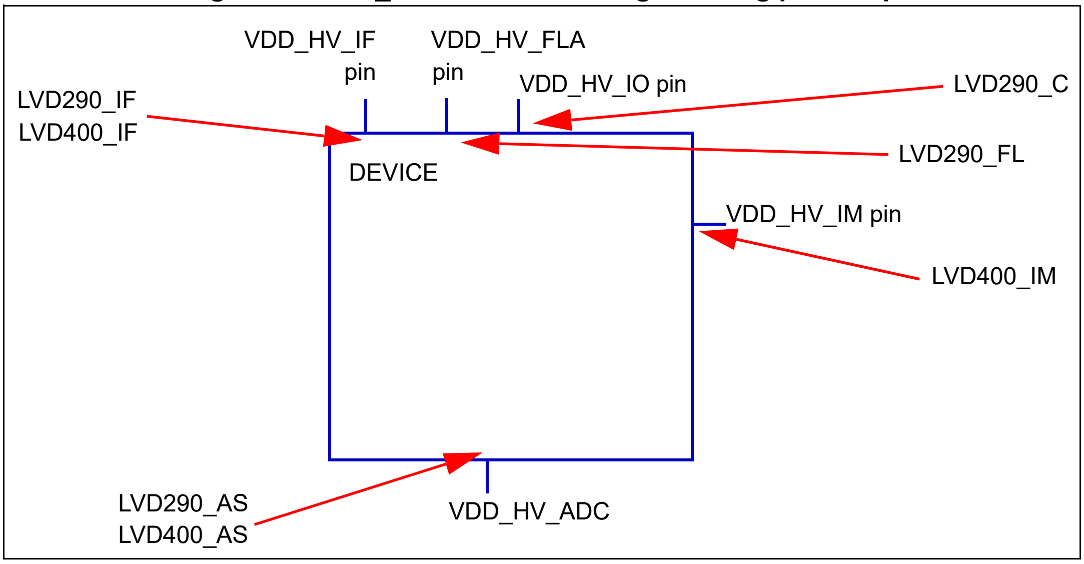
During POWERUP, the following high voltage MVDs are enabled:

- MVD270_C
- MVD270_FL

The VDD_HV conditions to exit POWERUP are the following:

- LVD290_C
- LVD290_FL

Figure 53. VDD_HV monitored voltages during power-up



The IRCOSC module starts initialization and provides the clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles.

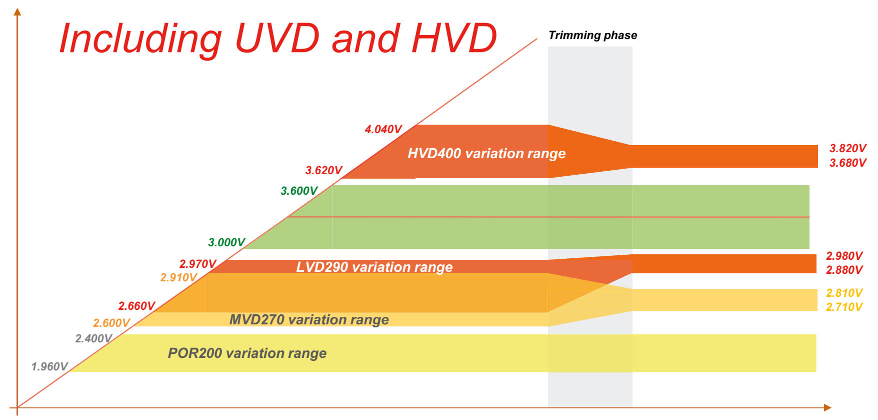
Device proceeds through the reset sequence through RGM phase PHASE0, PHASE1[DEST], PHASE2[DEST], PHASE3[DEST].

LVD/HVD modules are trimmed at the beginning of PHASE3[DEST]. Trimming is done by SSCM at low voltage, using the differential flash read accesses. After trimming is completed, SSCM waits for PMC acknowledgement before proceeding with reset sequence.

Configurable LVD/HVD modules are optionally enabled at the end of PHASE3[DEST].

Figure 54 provides the threshold variation of VDD_HV LVDs monitor during power-up sequence from POWERUP before and after trimming.

Figure 54. Threshold variation during power-up sequence on VDD_HV supply



The device needs to fulfill all conditions described in this section to recover.

10.6.2 Power-down sequence

In case LVDs/HVDs are configured to generate destructive reset , either LVD is below threshold or HVD is above threshold, the device enters PHASE0 phase.

Power-down sequence is actually entered as soon as the threshold of MVD094 and MVD270 is crossed. The device enters the POWERUP state. The internal power-on signal is asserted.

On power-on signal assertion, all modules reach their safe state.

Device supplies may then proceed to drop down to ground either through device leakage or external pull-down. During power-down, all supplies should comply with supply constraints.

10.6.3 Brown-out management

During brown-out, the device re-enters the POWERUP phase as soon as the threshold of either MVD094 or MVD270 is crossed. The internal power-on signal is asserted.

Note: The device will correctly start independently from any residual voltage.
At any rate there are corner cases on which the residual voltage above a certain limit can cause the MCU to get stuck indefinitely at the startup during the reset and boot phase. So to have a clean power-on sequence, it is recommended to satisfy the following two constraints at the same time once the device is exposed to brown-out: VDD_HV < 1.5V, VDD_LV < 0.4V.
For more details, refer to “Recommended power-up sequence” section in the Application note, SPC58xx hardware design guideline (AN4880).

Below MVD270 *threshold, the device is under POWERUP state. Above the* MVD270 *threshold, the device is into a RESET state as defined by Voltage monitor or external PORST pin or active state when all have been released.*

On internal power-on signal assertion, all analog modules reach their safe state.

10.6.4 Low voltage requirement during crank

The device can continue operation to the minimum input voltage during crank.

In order to proceed with execution during cranking and prevent device reset, it is important to correctly configure the high voltage LVDs.

During device switch-off the external LVDs monitoring may not be activated quickly enough (external assertion of the PORST pin) and hence the –10% level is crossed before POR is asserted. Malfunction is possible in this case, but reliability is not impacted. In particular the flash is not corrupted.

Internal LVDs that monitor the supply ensure that the flash content is protected.

11 Smart Stand-by Wake-up Unit (SSWU)

11.1 Overview

The Smart Standby Wake-Up Unit (SSWU) is aimed at further reducing the device power consumption by scheduling, on an RTC time basis, a sequence of user-programmable tasks, which are executed in STANDBY Mode without the CPU intervention.

These tasks are programmed in order to allow the following operations in STANDBY:

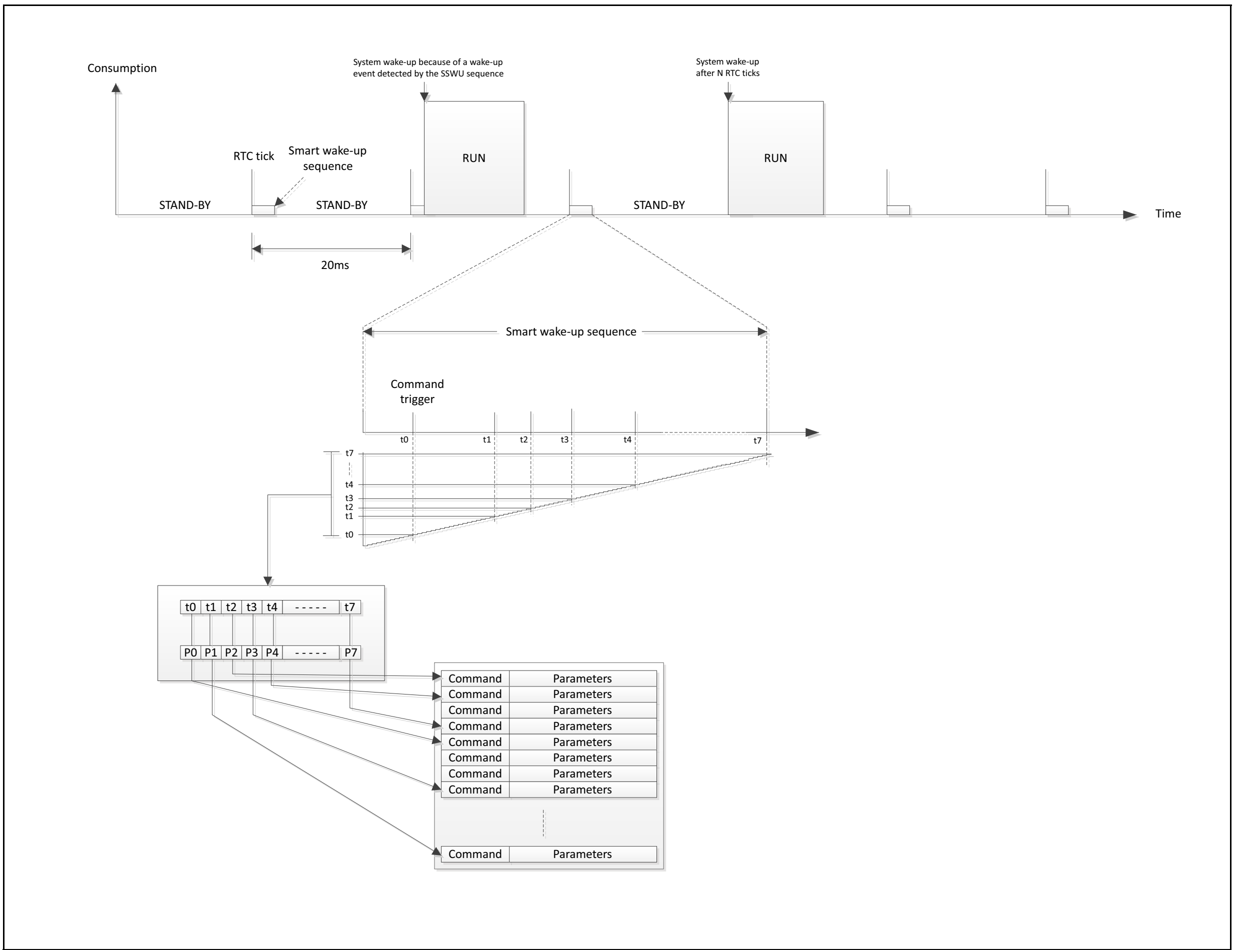
- Control output level
- Analog comparator versus voltage threshold or range
- Digital input state detector: In addition, it is possible to schedule a system wake-up based on a user-defined time interval.

The next sections explain the whole concept in details.

11.2 Functional description

Figure 55 shows the SSWU functionality.

Figure 55. Smart Wake-Up Sequence Functionality



The module is able to maintain the system mostly in stand-by and is eventually woken up based on the acknowledge of predifined wake-up event, which basically can be of two types:

- a time based wake-up event (WKUP1 line event)
- an asynchronous wake-up event (WKUP0 line event) detected during the SSWU sequence, which is periodically started based on the RTC tick (programming API period)

The time based wake-up event type is generated by the API/RTC IP, which can be programmed to generate it after a certain number of timer ticks.
The second asynchronous wake-up event type is generated by the SSWU block itself, as explained in the next sections.

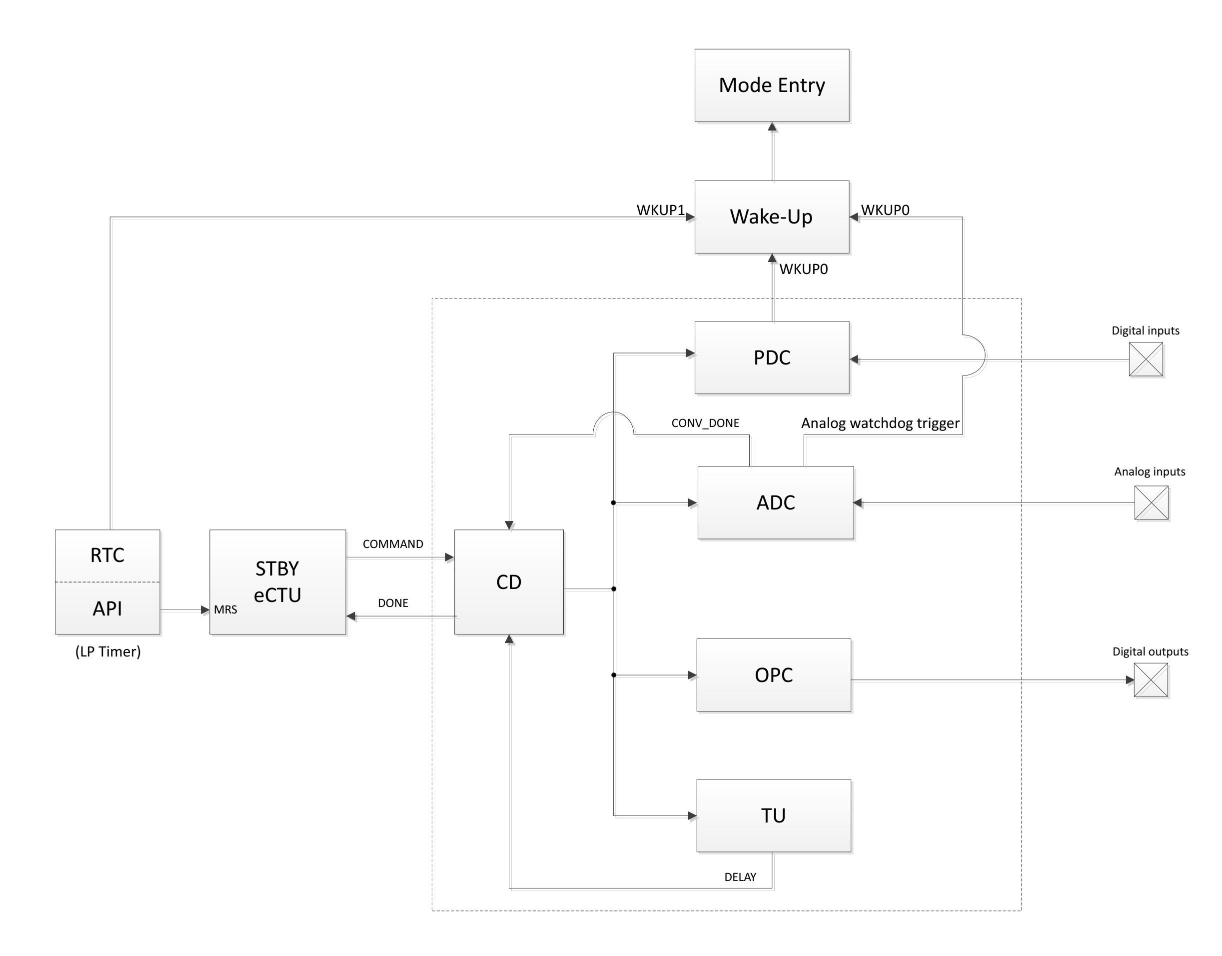
The SSWU sequence is a series of “commands” (plus some command-dependent parameters) generated by the Stand-by eCTU block based on a certain pre-programmed time schedule (t0, t1, etc., in the *Figure 55*). For each scheduled time event, it is possible to assign a command to be executed. As explained later on, it is possible to assign a sub-sequence of commands, which can be displaced with a programmable time delay. Each command is then interpreted by a command decoder, which is dispatching them to the target client.

11.3 Architecture

Once the device is in STANDBY Mode, a minimum set of IPs will be periodically woken-up. This allows a pre-programmed sequence of measurements to run, limiting the power consumption to the minimum requested for a body application.

Figure 56 shows the block diagram of the SSWU architecture.

Figure 56. Smart Wake-up Architecture



Hereafter is a list of the blocks in the picture.

- RTC = Real Time Clock
- STBY eCTU = Stand-by Enhanced Cross Triggering Unit
- CD = Command Decoder
- ADC = Analog Digital Converter (client)
- PDC = Port Digital Comparator (client)
- OPC = Output Pin Control block (client)
- TU = Timer Unit (client)

Hereafter is a brief description of each block, which is explained in details in the following chapters.

The RTC/API timers block is always powered-on in STANDBY and generates a time tick based on the low power clock source (LPRC or OSC32K) selected.

The RTC timer (mapped on the WKUP1 line) allows to periodically exit from STANDBY. The API timer (mapped on the WKUP0 line), instead, drives the starting of SSWU mechanism (refer to *Section 11.4*).

The Stand-by eCTU (Enhanced Cross Triggering Unit) is used to schedule the execution of a set of pre-programmed commands: it is the "engine" of the SSWU sequence.

The CD (Command Decoder) is used to decode the command generated by the Stand-by eCTU and dispatch them to the following client blocks:

- The ADC (Analog Digital Converter) is a dedicated, low-power analog to digital converter, used to generate an API wake-up event (WKUP0) in case the converted value is above or below a certain threshold.
- The OPC (Output Pin Control block) block is used to set the output level of a digital port pin, in order to let the customer open/close some external switches.
- The PDC is used to trigger an API wake-up event (WKUP0) when the level of a pin is a certain value for a specific time.
- The TU is used to generate a programmable delay between different Stand-by eCTU command executed within the same command sequence.

The Wake-up unit collects the wake-up triggers (eventually provided by the ADC or PDC client) and generates a wake-up event (WKUP0) for the Mode Entry block.

The Mode Entry block is then responsible to change the system execution mode (from STANDBY to Run).

In STANDBY mode, only RTC is powered on, while all other blocks are off. At the RTC tick, the hardware switches on the 16 MHz IRCOSC and wakes up the other blocks.

11.4 Clocking

Figure 57 shows the SSWU clocking architecture.

The RTC evolves with the low power clock, which can be selectable between 32 kHz or 128 kHz.

Note: The 1024 kHz clock signal from slow internal RC oscillator (LPRC) is divided by 8 before applied as 128 kHz clock to RTC.

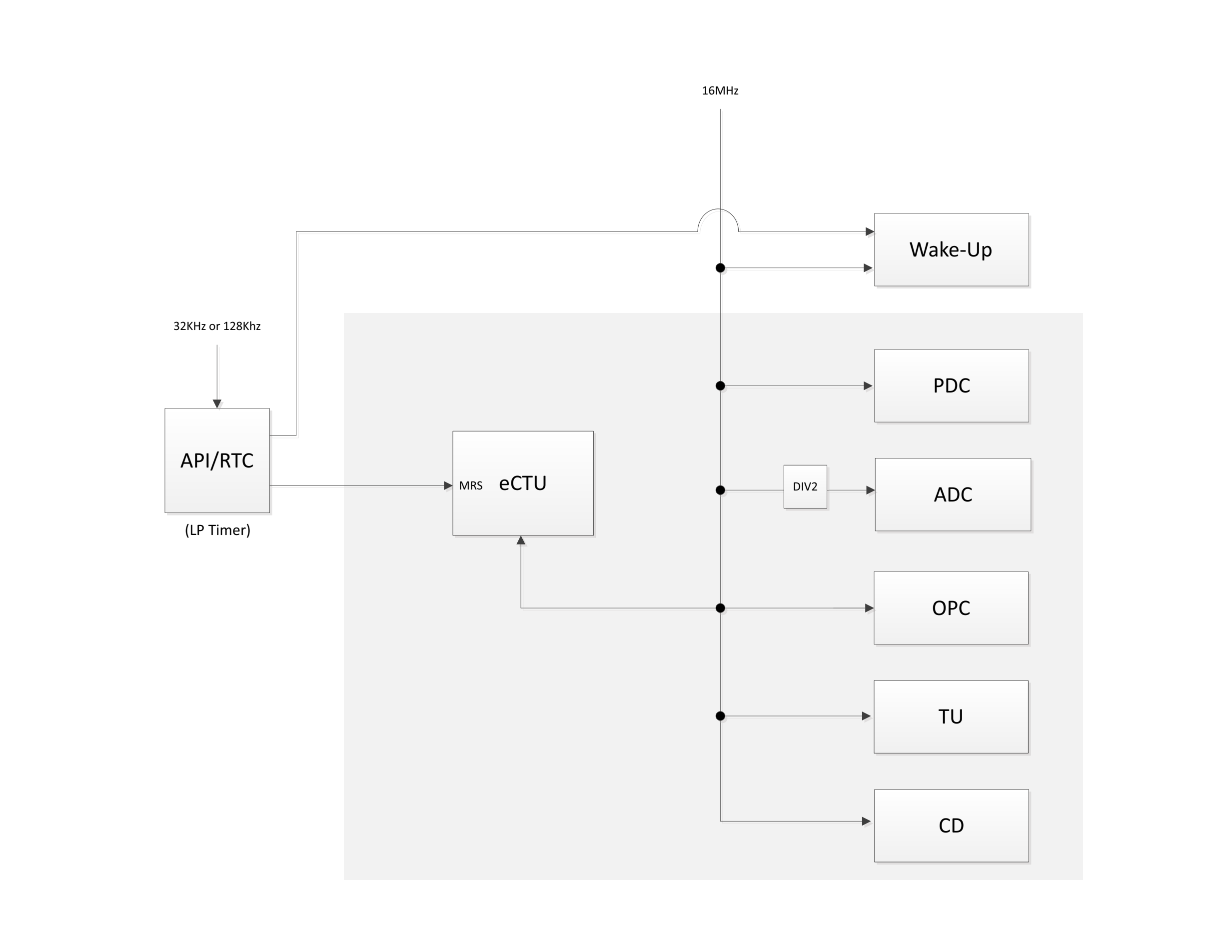
The SSWU IPs evolves with the internal 16 MHz IRCOSC.

A prescaler is embedded in the TU block in order to let the customer program the delay resolution (refer to *Section 11.6.6: TU*).

Note: Even if IRCOSC is configured off in STANDBY through the ME_STANDBY0_MC register, it will be automatically turned-on once SSWU is activated. The IRCOSC will be automatically turned off after SSWU Stop command is executed.

The ADC IP runs at 8MHz (IRCOSC clock divided by 2).

Figure 57. SSWU clock architecture



11.5 Memory map and register descriptions

11.5.1 Memory map

This section provides the memory map of the SSWU. For double buffering and synchronization refer to *Section 11.5.13: Double buffered registers*.

11.5.1.1 TGS registers

Table 117. TGS registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0000	Trigger Generator Subunit Input Selection Register (TGSISR)	Yes	TGSISR_RE	0x0000 0000
0x0004	Trigger Generator Subunit Control Register (TGSCR)	Yes	MRS	0x0000 0000
0x0008	Trigger Compare Register 0 (TCR0)	Yes	MRS	0x0000 0000

Table 117. TGS registers (continued)

Address offset	Register	Double buffered	Synchronization	Reset value
0x000C	Trigger Compare Register 1 (TCR1)	Yes	MRS	0x0000 0000
0x0010	Trigger Compare Register 2 (TCR2)	Yes	MRS	0x0000 0000
0x0014	Trigger Compare Register 3 (TCR3)	Yes	MRS	0x0000 0000
0x0018	Trigger Compare Register 4 (TCR4)	Yes	MRS	0x0000 0000
0x001C	Trigger Compare Register 5 (TCR5)	Yes	MRS	0x0000 0000
0x0020	Trigger Compare Register 6 (TCR6)	Yes	MRS	0x0000 0000
0x0024	Trigger Compare Register 7 (TCR7)	Yes	MRS	0x0000 0000
0x0028 - 0x0047	Reserved			
0x0048	TGS Counter Compare Register (TGSCCR)	Yes	MRS	0x0000 0000
0x004C	TGS Counter Reload Register (TGSCRR)	Yes	MRS	0x0000 0000

11.5.1.2 SU registers

Table 118. SU registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0100	Commands List Control Register 1 (CLCR1)	Yes	MRS	0x0000 0000
0x0104	Commands List Control Register 2 (CLCR2)	Yes	MRS	0x0000 0000
0x0108 - 0x010F	Reserved			
0x0110	Trigger Handler Control Register 1 (THCR1)	Yes	MRS	0x0000 0000
0x0114	Trigger Handler Control Register 2 (THCR2)	Yes	MRS	0x0000 0000
0x0118 - 0x011F	Reserved			
0x0120 - 0x019C	Command List Register 1-32 (CLR1-CLR32)	Yes	MRS	0x0000 0000
0x01A0 - 0x031F	Reserved			

11.5.1.3 Other eCTU registers

Table 119. Other eCTU registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0500 - 0x050F	Reserved			
0x0510	CTU Control Register (CTUCR)	NO	handshake	0x0000 0000
0x0514 - 0x054B	Reserved			

11.5.2 Trigger generator subunit input selection register (TGSISR)

Offset: 0x000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	I0_FE	I0_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 58. Trigger generator subunit input selection register (TGSISR)

Table 120. TGSISR field descriptions

Field	Description
30 I0_FE	ctu_trg_in[0] Falling edge Enable 0 disabled 1 enabled
31 I0_RE	ctu_trg_in[0] Rising edge Enable 0 disabled 1 enabled

Note: The IO_RE bit should be set in order to enable the SSWU mechanism.

11.5.3 Trigger generator subunit control register (TGSCR)

Offset: 0x0004 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PRES		Reserved					TGS_M
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 59. Trigger generator subunit control register (TGSCR)

Table 121. TGSCR field descriptions

Field	Description
24:25 PRES	TGS prescaler selection bits 00 1 01 2 10 3 11 4
31 TGS_M	Trigger Generator Subunit Mode 0 Triggered Mode 1 Sequential Mode **Note:** the only mode for eCTU STANDBY is the triggered mode, so the user has to keep TGS_M = 0.

11.5.4 Trigger compare register 0–7 (TCR0–TCR7)

There are 8 TCRx registers to support 8 triggers.

Offset: 0x008–0x0024 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TxCR_value															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 60. Trigger compare register 0–7 (TCR0–TCR7)

Table 122. TCR0–TCR7 field descriptions

Field	Description
16:31 TxCR_value	Value to be compared with the counter value to generate the trigger.

11.5.5 TGS counter compare register (TGSCCR)

Offset: 0x0048 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TGSCCR_value															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 61. TGS counter compare register (TGSCCR)

Table 123. TGSCCR field descriptions

Field	Description
16:31 TGSCCR_value	This register is used to stop the counter when TGSCCR_value is reached.

11.5.6 TGS counter reload register (TGSCRR)

Offset: 0x004C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TGSCRR_value															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 62. TGS counter reload register (TGSCRR)

Table 124. TGSCRR field descriptions

Field	Description
16:31 TGSCRR_value	This register is used to reload the counter when MRS is one.

11.5.7 Commands list control register 1 (CLCR1)

Offset: 0x0100 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T3_INDEX					0	0	0	T2_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T1_INDEX					0	0	0	T0_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 63. Commands list control register 1 (CLCR1)

Table 125. CLCR1 field descriptions

Field	Description
3:7 T3_INDEX	Points to the (1st command address-1) for Trigger 3.
11:15 T2_INDEX	Points to the (1st command address-1) for Trigger 2.
19:23 T1_INDEX	Points to the (1st command address-1) for Trigger 1.
27:31 T0_INDEX	Points to the (1st command address-1) for Trigger 0.

Note: First command for any trigger will start from CLR < Tx_INDEX + 1 > register.

11.5.8 Commands list control register 2 (CLCR2)

Offset: 0x0104 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T7_INDEX					0	0	0	T6_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T5_INDEX					0	0	0	T4_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 64. Commands list control register 2 (CLCR2)

Table 126. CLCR2 field descriptions

Field	Description
3:7 T7_INDEX	Points to the (I[st] command address-1) for Trigger 7.
11:15 T6_INDEX	Points to the (I[st] command address-1) for Trigger 6.
19:23 T5_INDEX	Points to the (I[st] command address-1) for Trigger 5.
27:31 T4_INDEX	Points to the (I[st] command address-1) for Trigger 4.

Note: First command for any trigger will start from CLR < Tx_INDEX + 1 > register.

11.5.9 Trigger handler control register 1 (THCR1)

Offset: 0x0110 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T3_E	0	0	0	T3_CMDE	0	0	0	T2_E	0	0	0	T2_CMDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T1_E	0	0	0	T1_CMDE	0	0	0	T0_E	0	0	0	T0_CMDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 65. Trigger handler control register 1 (THCR1)

Table 127. THCR1 field descriptions

Field	Description
3 T3_E	Trigger 3 enable 0 disabled 1 enabled
7 T3_CMDE	Trigger 3 Command output enable 0 disabled 1 enabled
11 T2_E	Trigger 2 enable 0 disabled 1 enabled
15 T2_CMDE	Trigger 2 Command output enable 0 disabled 1 enabled

Table 127. THCR1 field descriptions (continued)

Field	Description
19 T1_E	Trigger 1 enable 0 disabled 1 enabled
23 T1_CMDE	Trigger 1 Command output enable 0 disabled 1 enabled
27 T0_E	Trigger 0 enable 0 disabled 1 enabled
31 T0_CMDE	Trigger 0 Command output enable 0 disabled 1 enabled

11.5.10 Trigger handler control register 2 (THCR2)

Offset: 0x0114 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T7_E	0	0	0	T7_CMDE	0	0	0	T6_E	0	0	0	T6_CMDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T5_E	0	0	0	T5_CMDE	0	0	0	T4_E	0	0	0	T4_CMDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 66. Trigger handler control register 2 (THCR2)

Table 128. THCR2 field descriptions

Field	Description
3 T7_E	Trigger 7 enable 0 disabled 1 enabled
7 T7_CMDE	Trigger 7 Command output enable 0 disabled 1 enabled
11 T6_E	Trigger 6 enable 0 disabled 1 enabled

Table 128. THCR2 field descriptions (continued)

Field	Description
15 T6_CMDE	Trigger 6 Command output enable 0 disabled 1 enabled
19 T5_E	Trigger 5 enable 0 disabled 1 enabled
23 T5_CMDE	Trigger 5 Command output enable 0 disabled 1 enabled
27 T4_E	Trigger 4 enable 0 disabled 1 enabled
31 T4_CMDE	Trigger 4 Command output enable 0 disabled 1 enabled

11.5.11 Commands list register 1–32 (CLR1–CLR32)

The number of CLRx supported is 32.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PARAMS							0	0	0	0	0	0	CLIENT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 67. Commands list register 1–32 (CLR1–CLR32)

Table 129. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.

Table 129. CLR1–CLR32 field descriptions (continued)

Field	Description
17:23 PARAMS	Client parameters Its structure varies according to selected client.
30:31 CLIENT	Target client of the command 00 ADC 01 OPC 10 PDC 11 TU

11.5.11.1 Commands list register 1–32 (CLR1–CLR32) - ADC client

The number of CLRx supported is 32.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	EXTCH_SEL	CHANNEL						0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 68. Commands list register 1–32 (CLR1–CLR32)

Table 130. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.
17 EXTCH_SEL	Internal / External channel selection bit 0 Internal channel 1 External channel
18:23 CHANNEL	Channel number Refer to *Table 136: ADC channel selection* for details.
30:31 CLIENT	Target client of the command 00 ADC

11.5.11.2 Commands list register 1–32 (CLR1–CLR32) - OPC client

The number of CLRx supported is 32.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PATTERN				GROUP			0	0	0	0	0	0	0	1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 69. Commands list register 1–32 (CLR1–CLR32)

Table 131. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.
17:20 PATTERN	Pattern to be applied to selected group of four OPC channels.
21:23 GROUP	OPC channels selection 000 Output OPC channels 1-3 001 Output OPC channels 4-7 01x Reserved
30:31 Client	Target client 01 OPC

11.5.11.3 Commands list register 1–32 (CLR1–CLR32) - PDC client

The number of CLRx supported is 32.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	COMMAND		PDC_CHANNEL					0	0	0	0	0	0	1	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 70. Commands list register 1–32 (CLR1–CLR32)

Table 132. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.
17:18 COMMAND	Command to execute on selected channel 00 CMP_0 (compare input channel with level '0') 01 CMP_1 (compare input channel with level '1') 10 CHECKOUT (evaluate comparison results) 11 Reserved
19:23 PDC_CHANNEL	PDC channel number
30:31 CLIENT	Target client of the command 10 PDC

11.5.11.4 Commands list register 1–32 (CLR1–CLR32) - TU client

The number of CLRx supported is 32. Refer to *Section 11.6.6* for more details about TU function.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	DELAY							0	0	0	0	0	0	1	1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 71. Commands list register 1–32 (CLR1–CLR32)

Table 133. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.
17:23 DELAY	Programmable delay before next command starts Time delay = DELAY * $T_{RC16MHz}$* TU_{PRESC}(1)
30:31 CLIENT	Target client of the command 11 TU

1. TU_{PRESC} is defined in the SSWU_CTRL_REG register. Please refer to *Chapter 54: Power management controller digital interface (PMC_Dig)*, *Section 54.5.28: SSWU CTRL REG (SSWU_CTRL_REG)*.

11.5.12 Cross triggering unit control register (CTUCR)

Offset: 0x0510 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	Reserved(1)	Reserved(1)	Reserved(1)	Reserved(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CTU_FSM_RESET	Reserved	0	CGRE	FGRE	Reserved(1)	GRE	TGSISR_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Never write 1 to this bit.

Figure 72. Cross triggering unit control register (CTUCR)

Table 134. CTUCR field descriptions

Field	Description
24 CTU_FSM_RESET	CTU state machine reset when written ‘1’. Writing a one has effect on the module, but it always reads as zero. **Note:** It is possible that CTU state machine is waiting for end of SSWU command, and due to any fault, it does not arrive. Therefore, this bit should be used to reset the CTU state machine before each standby entry.
27 CGRE	Clear GRE (1)

Table 134. CTUCR field descriptions (continued)

Field	Description
28 FGRE	Flag GRE [(1)]
30 GRE	General Reload Enable [(1)] 0 disabled 1 enabled
31 TGSISR_RE	TGS Input Selection Register Reload Enable. Is set by the user and reset automatically when its reloaded.

1. Refer to *Section 11.5.13* for details.

11.5.13 Double buffered registers

Some registers of the SSWU are double buffered. When MRS is set, the data programmed in a register on register interface clock domain is transferred to the SSWU kernel clock domain. To avoid error condition due to a MRS signal that arrives when a write operation is not fully completed, the following 2 bits are used:

1. GRE (General Reload Enable)
2. FGRE (Flag General Reload disable)

The reload of all double-buffered registers is enabled by setting the GRE bit in the CTU Control Register in order to guarantee the consistency of all double-buffered registers. The user must ensure that the trigger command list is updated before MRS occurs. If the update is not performed, the previous values of all double-buffered registers remain active.

All the double-buffered registers use the same bit (GRE) to enable the reload when the MRS occurs. This bit is used to signal that the user has updated the registers and so a reload can be performed without any problem. The MRS bit is R/S (Read/Set). If the bit is 1 the reload can be performed, while if this bit is 0, the reload is not performed. A correct reload resets this bit. It is also possible to reset this bit by software setting CGRE (Clear GRE) bit. This bit can be reset in a software way but, anyway, it is reset when GRE bit is at zero.

In order to verify if a reload error occurs, FGRE (Flag GRE bit in the CTU Control Register) bit is used. When one of the double-buffered registers is written, FGRE flag is set to 1 and it is reset by a correct reload. When the MRS occurs while FGRE = 1 and GRE = 1, a correct reload is performed (because all intended registers have been updated before the MRS occurs).

- If FGRE = 1 and GRE = 0, a reload is not performed, the error flag (MRS_RE) is set (in this case at least one register was written but the update has not been ended before the MRS occurrence).
- If FGRE = 0, it is not necessary to perform a reload because all the double-buffered registers are unchanged.
- If GRE = 1 with FGRE = 0, a MRS_RE is generated on the next master reload.

TGSISR_RE bit is used to synchronize TGSISR register. When the user set TGSISR_RE, the value written in the TGSISR register is used.

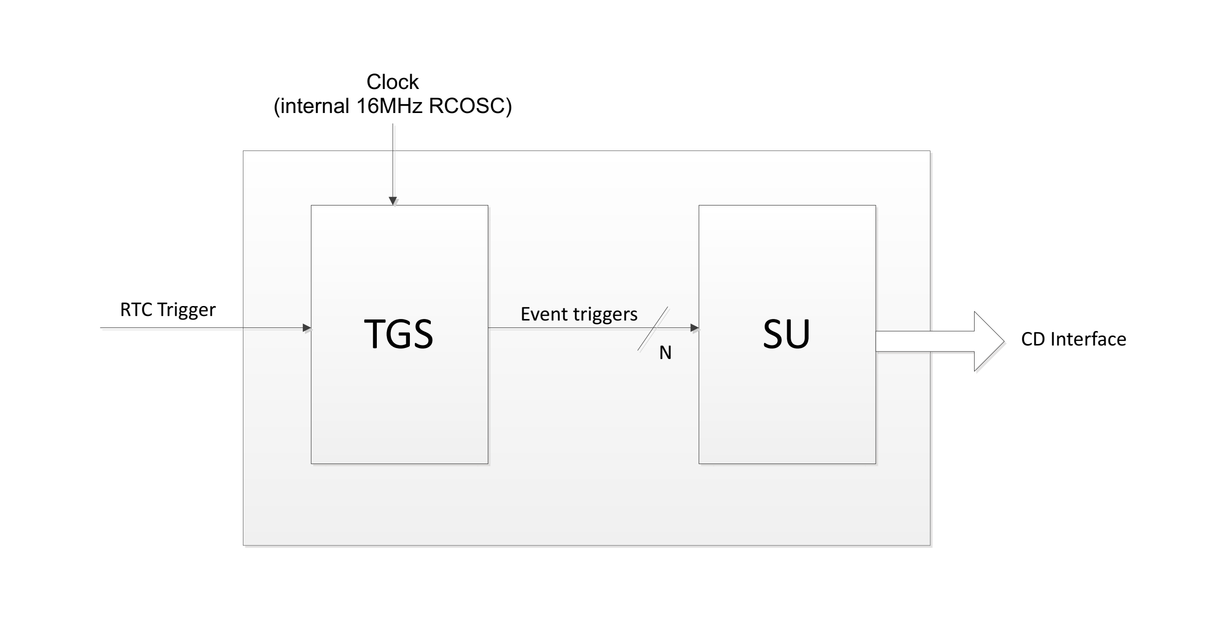
11.6 SSWU module descriptions

11.6.1 Stand-by eCTU

Figure 73 shows the Stand-by eCTU block diagram. Basically it is composed of two blocks:

- TGS (Trigger Generator Sub-unit), which is responsible to generate a set of triggers based on some pre-programmed time schedules
- SU (Scheduler Unit), which receives the trigger and starts the associated command sequence

Figure 73. Stand-by eCTU Block Diagram



The figure shows that the output of the Stand-by eCTU is directed to the CD interface. The CD block of the SSWU decodes the CTU command and dispatches it to the various clients (OPC, PDC, ADC, TU).

11.6.1.1 TGS block

Figure 74 shows the Stand-by eCTU architecture and functionality.

As shown in the figure, the TGS contains a 16-bit counter unit (evolving with the internal slow clock) and a set of comparators (TxCRV). Each comparator determines one time event point (tx in *Figure 55*).

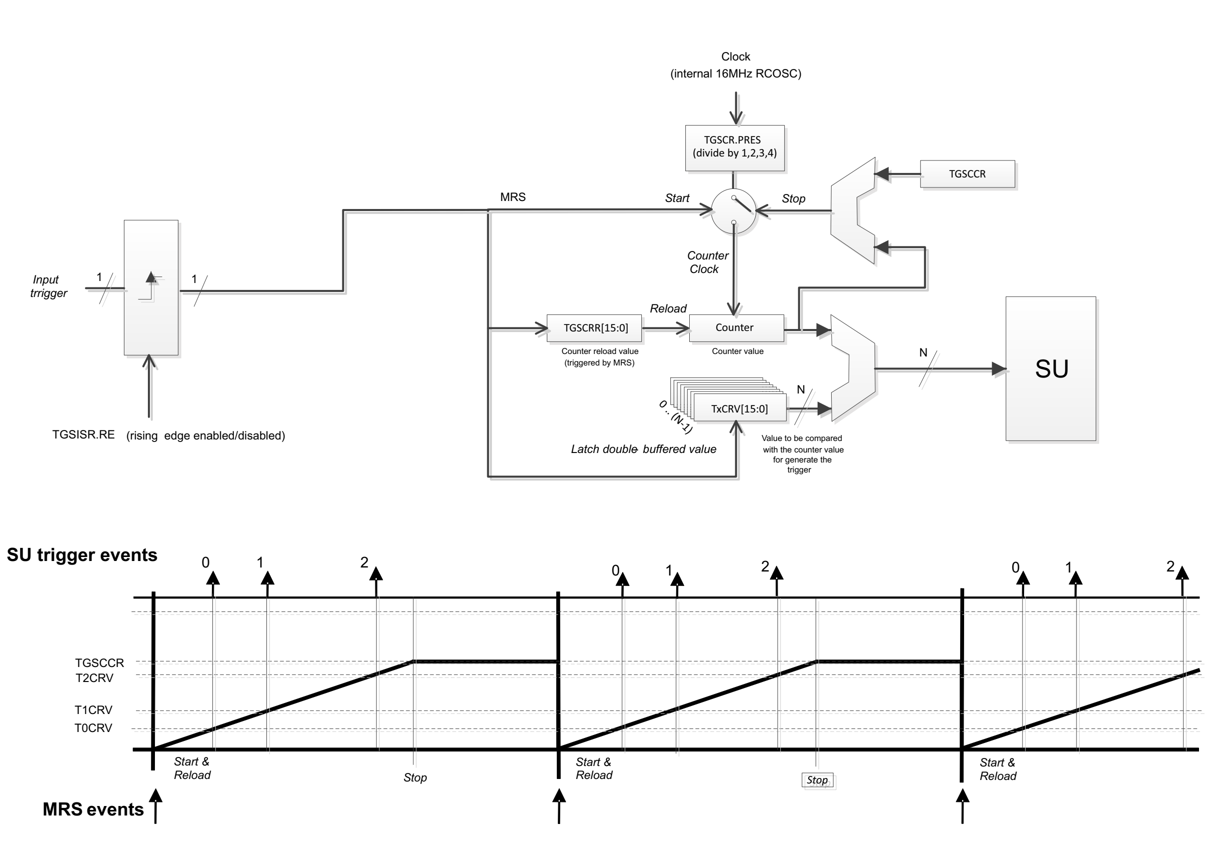
The MSR (Master Reload Signal) is used to reset and start the counter. The reset value is stored in the TGSCRR register. The MRS is generated via HW trigger (RTC tick).

When the counter reaches one of the values programmed in the TxCRV comparators, an internal trigger is generated to the SU block. Note that is it possible to program TGSCCR register so that the counter will be stopped as soon as it reaches a certain value.

The waveform in *Figure 74* shows an example of three SU trigger generations, based on T0CRV, T1CRV, and T2CRV comparators. Note that, the TGS internal counter is started by a MRS event and reloaded with the '0' value. Also the TGSCCR register is used to stop the counter when it reaches the value programmed in this register.

Caution: When the SSWU sequence is generated with several SU triggers, the user must insure the SSWU command is finished before triggering a new one.

Figure 74. TGS Architecture and Functionality (triggered mode)



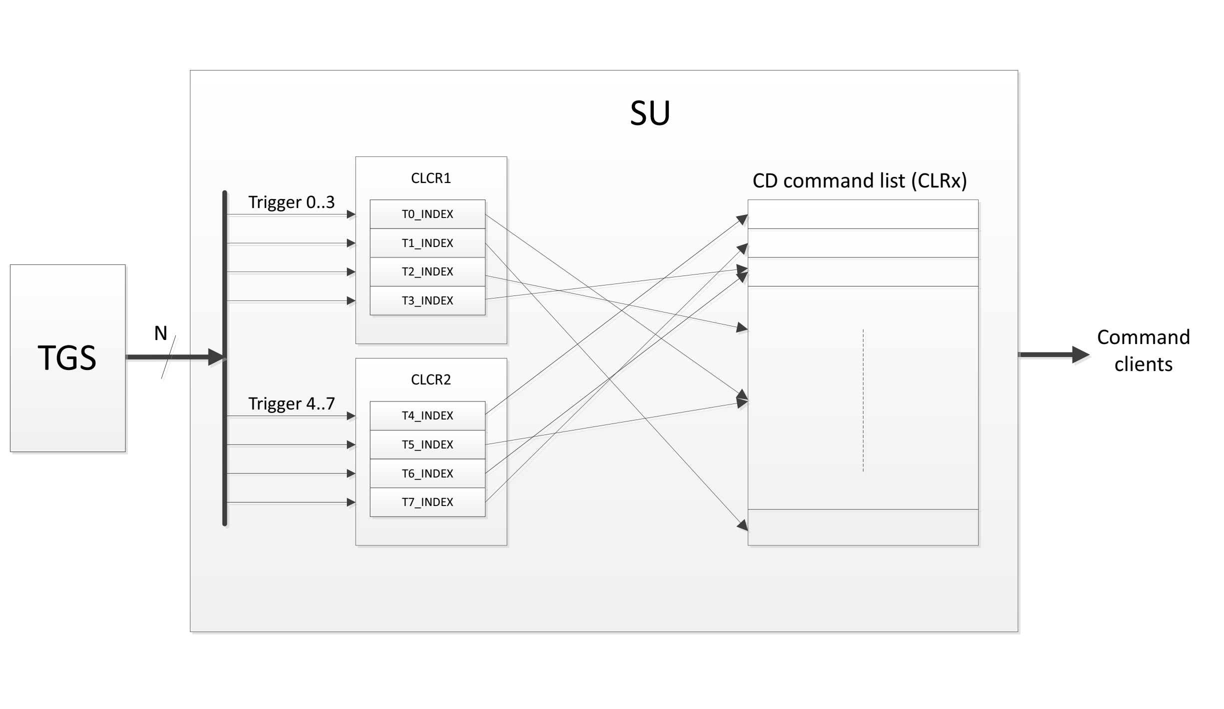
11.6.1.2 SU block

Figure 75 shows the SU block functionality. Its function is to receive the internal SU trigger and activate a sequence of commands, pre-programmed in a command table [d].

In fact, each trigger from the TGS block is associated to a programmable pointer, which determines, within the SU command table, the begin of the sequence of commands.

d. The command table length is 32

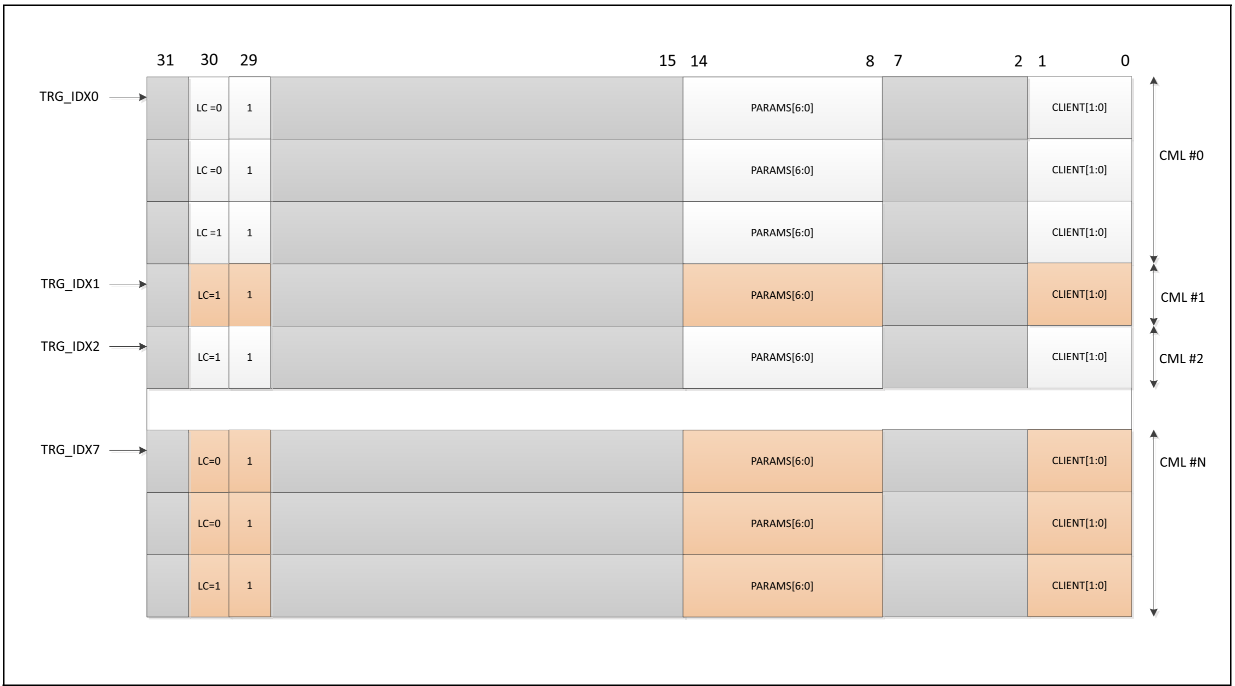
Figure 75. SU Functionality



11.6.1.3 Stand-by eCTU command structure

The command structure is specified in *Figure 76*.

Figure 76. SU Command Structure



Hereafter is a description of the command fields.

- LC (last Command): this flag is used to specify whether the command being executed is the last of the sequence started by a SU trigger. *Figure 76* shows several examples: the command sequence triggered by TRG_IDX0 is composed of 3 commands, while the sequence triggered by TRG_IDX1 is actually one single command.
- EC (bit 29): always set to 1
- PARAMS: specify the command parameters
- CLIENT: specify the client ID

Starting from the first command to be executed, as programmed in the pointer associated to a specific SU trigger, each command is executed after the previous one is completed. The “LC” command flag determines whether the current command is the last one to be executed for the sequence of commands being executed.

11.6.1.4 Stand-by eCTU command sequence summary

As a summary, the command sequence is driven by:

- the TGS sequence of triggers, where each SU trigger is generated on a programmable time-basis,
- the SU command sequence, activated by a SU trigger.

The Stand-by eCTU commands are executed in a row till the one marked as "LC = 1". The Stand-by eCTU executes the next command as soon as it is acknowledged that the previous command was completed. This is represented in *Figure 56* by the DONE feedback signal from the CD block to the Stand-by eCTU block.

The execution time between each command depends on the command itself.

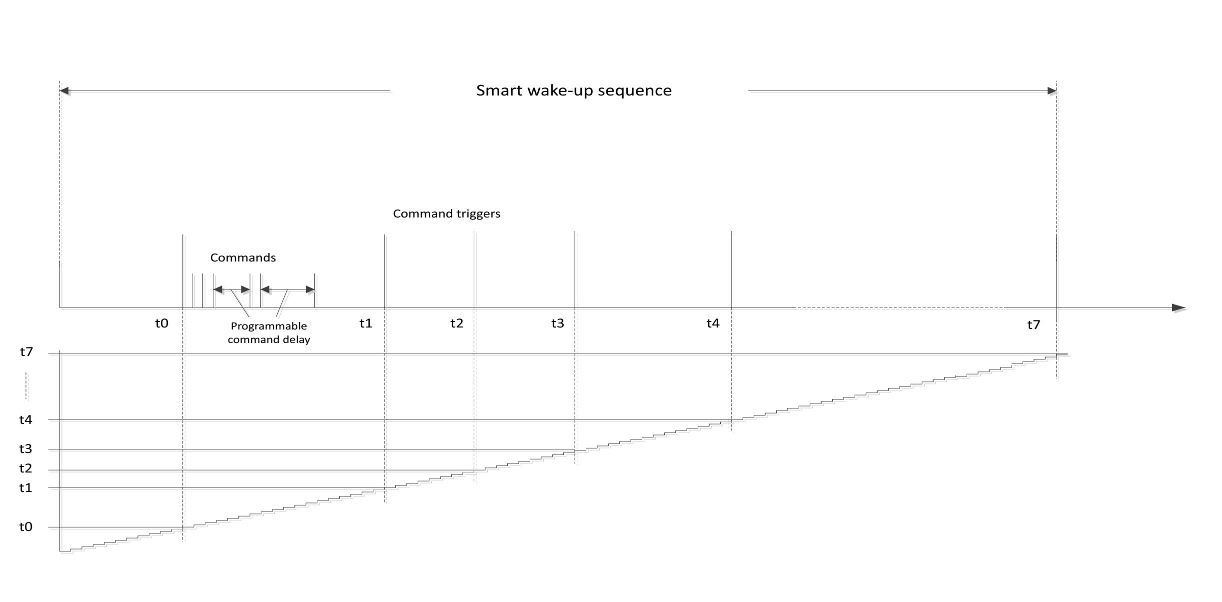
A command associated with an ADC conversion lasts till the ADC has acknowledged the Stand-by eCTU that the conversion was completed. This is represented in *Figure 56* by the CONV_DONE feedback signal from the ADC block to the CD block.

Other commands (for example, the one used to set a digital port pin level) have no “intrinsic” duration. Because of this, the “TU” block has been introduced, in order to be able to set a programmable delay between “immediate” commands. This is represented in *Figure 56* by the DELAY feedback signal from the TU block to the CD block.

As a result, shown in *Figure 77*, the time when a command shall be executed can be controlled in two ways:

- the value of the TxCVR comparators (t0, t1, etc. in *Figure 77*)
- the TU delay times (“programmable command delay” in *Figure 77*)

Figure 77. Programmable Command Delay



11.6.2 CD (Command Dispatcher)

The Stand-by eCTU commands contain two main fields:

- CLIENT
- PARAMS

The information in these two fields is decoded by the CD block and dispatched to the target clients (refer to *Table 135*).

The PARAMS field of the command is interpreted according to the type of client being activated.

Table 135. Smart Wake-up Sequence command

CLIENT[1:0]	Block	PARAMS[6:0]
0	ADC	Refer to *Table 136*
1	OPC	Refer to *Table 137*
2	PDC	Refer to *Table 138*
3	TU	Wait time

Table 136. ADC channel selection

PARAMS[6]	PARAMS[5:0][(1)]
ADC Internal or External channel selection	ADC channel number

1. Refer to *Section 7.3.12: SSWU configuration* for the association between the parameter values and the ADC channels

Table 137. OPC Command

PARAMS bits	Field name	Value	Description
[6:3]	PATTERN	Any	Pattern to be applied to the selected 4 channel group
[2:0]	GROUP	000	Select channels 0..3
		001	Select channels 4..7
		010–111	Reserved

Table 138. PDC Commands

PARAMS bits	Field name	Value	Description
[6:5]	COMMAND	00	CMP 0
		01	CMP 1
		10	CHECKOUT
		11	Reserved
[4:0]	PDC channel	Any	PDC channel number

Table 139. TU Commands

PARAMS bits	Field name	Value	Description
[6:0]	DELAY	Any	Programmable delay

11.6.2.1 PDC channel handling

The SSWU is able to drive 14 PDC channels: PDC0-PDC4, PDC7-PDC16, PDC5 and PDC6 are reserved. The PDC channel number is selected by the PDC command parameter (refer to *Table 138*).

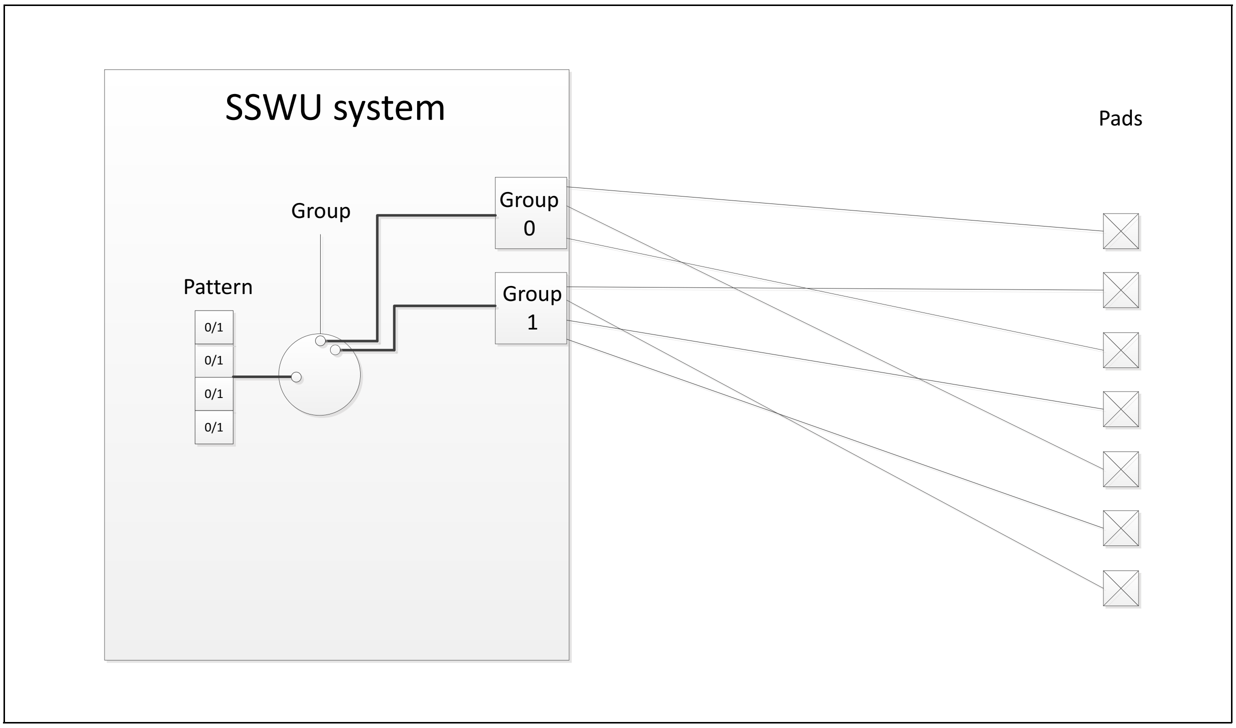
The connection between each PDC channel and the corresponding pad is hard-wired as specified in the Microsoft Excel pinout file (I/O Signal Table).

11.6.2.2 OPC channel handling

The SSWU is able to drive up to 7 OPC channels. OPC0 is reserved.

The channels are divided into 2 groups of 4 channels each. The “GROUP” command parameter is used to select the group of channels to be operated (refer to *Table 137*). The 4 channels belonging to the selected group are driven in parallel: the 4 channels will be set according to the value specified in the “PATTERN” command parameter. The connection between each OPC channel and the corresponding pad is hard-wired as specified in the Microsoft Excel pinout file (I/O Signal Table).

Figure 78. SSWU OPC connections



11.6.3 ADC

The ADC block is a reduced, low-power version of the Successive Approximation Register Analog-to-Digital Converter called 10-bit SARADC_STDBY. Its main functionality is to compare the converted value towards a watchdog, and trigger a wake-up event in case the value is out of the intended range.

The watchdog thresholds (low and upper limits) are linked (that is pre-programmed during the RUN mode) to each ADC channel, so that a specific channel has its own threshold limit.

The ADC configuration comprises:

- 24 internal channels
- 4 different watchdog threshold values
- 8 external channels

More details about the ADC configuration are given in the *Chapter 38: Analog-to-Digital Converters (ADC) Configuration*.

11.6.3.1 ADC Channel Selection

The selection of internal or external channel group is done through the PARAMS[6] bit.

In case the internal channel group is selected (PARAMS[6] = 0), the internal channel number depends on the value specified by the PARAMS[5:0] bits. Because the internal channels are statically connected to a specific analog input pin, the channel selection is also a selection vs. which analog input pins will be used (refer to *Section 7.3.12: SSWU configuration*).

In case the external channel group is selected (PARAMS[6] = 1), the value of PARAM[5:0] represents the external channel number, starting from the offset 128, for example if

PARAM[5:0] = 0, then the external channel 128 is selected; if PARAM[5:0] = 1, then the external channel 129 is selected, and so on (refer to *Section 7.3.12: SSWU configuration*). The selected external channel number is then re-mapped on an internal channel through the ECMICR registers, which contains a set of fields, one for each external channel, where the linked internal channel can be specified. This association has to be done before entering into the stand-by mode. In this way, it is possible to link different external channels to the same internal channel.

11.6.3.2 ADC Watchdog Threshold Selection

A voltage measurement threshold (upper and lower limit) can be selected for each channel. The WTHRHRLx registers define a pool of watchdog threshold values, which can be referenced for each channel through the ICWSELx and ECWselx registers for the internal or external channel respectively.

In case the same analog input pin is used to measure the same voltage range with the same voltage threshold, then an internal channel is used (PARAMS[6] = 0). In this case, the external channels are grouped in slice of 8 channels. Each slice is selected by PARAMS[5:3] and can be associated to an internal channel. PARAMS[2:0] selects which external channel is used, among the 8 channel of a slice.
In this scenario, the configuration is:

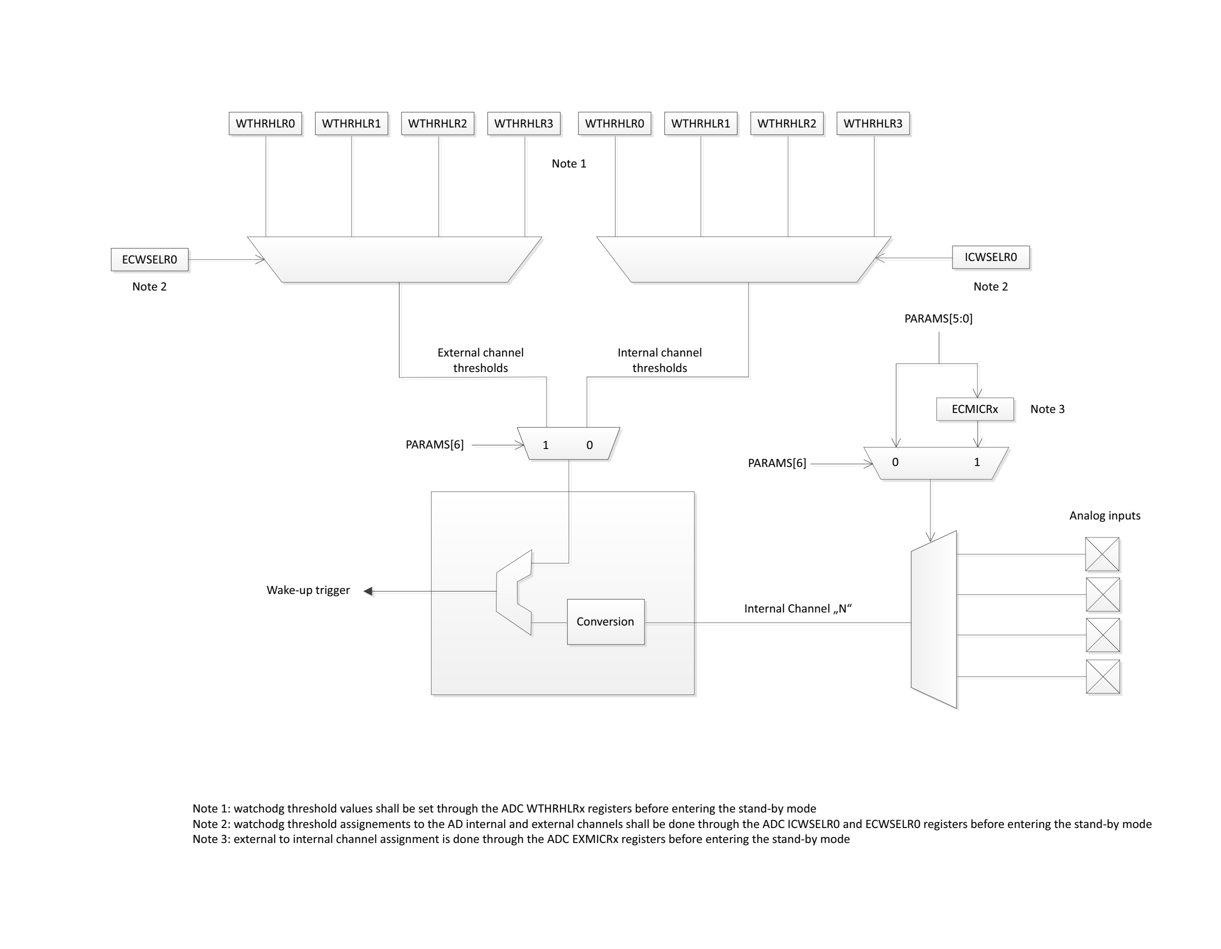
- the internal channel value is fixed by the PARAMS[5:0] value
- the threshold is chosen by setting the ICWSEL register before entering the stand-by mode

In case the same analog input pin is used to measure the different voltage range with the same voltage threshold, then an external channel is used (PARAMS[6] = 1). In this scenario, the configuration is:

- the external channel value is fixed by the PARAMS[5:0] value
- the threshold is chosen by setting the ECWSEL register before entering the stand-by mode
- the internal channel (the pin to be used) linked to the chosen external channel is defined by the ECMICR register

Figure 79 represents in graphical form the above description.

Figure 79. ADC channel selection



11.6.4 OPC

The OPC block is used to drive the 7 digital output pins to a desired level.

The intended functionality is to open/close external switches.

The Stand-by eCTU command parameter contains the OPC channel and level.

The OPC channels are controlled in groups of 4 channels, but, it is possible to mask any one of these channels, in order to retain the value set in RUN mode on any pad not intended to be used for the SSWU.

The mechanism to achieve this is to use the SIUL SCR bits to mask the OPC channel driving to the pad (refer to *Section 16.2.2.11: Soc Configuration Register 0 (SIUL2_SCR0)*). In order to drive an OPC channel, the SW shall configure the associated SCR bit before entering in stand-by mode.

The set of pads used to control external logic during the Stand-by have the following behavior:

- Upon entering the stand-by mode, the pad values corresponding to OPC channels are maintained.
- That means if the pad is used for OPC functionality during STANDBY mode, then on the subsequent STANDBY mode entry, the PAD value is restored to the value during the previous STANDBY mode exit.
- However, if the PAD is not used for OPC functionality during STANDBY, then the same value as in RUN mode is maintained upon entering STANDBY mode.
- SIUL_SCR0.OPC_MASK configuration for any OPC channel should remain same during the application. Otherwise, the value on OPC channel on STANDBY entry may have some undesired transitions.
- During stand-by mode, it is possible to change the value of those pads thanks to OPC programming.
- Upon exiting from stand-by, the value of the pads connected to OPC channels can also be maintained. This avoids glitches during the transition between stand-by and run mode. For this purpose, the SIUL2_SCR0 register must be programmed before entering stand-by.

Note: Note that the pads value and direction can still be changed during the RUN mode, but this requests to write a bit within the SIUL_SCR0 register to “1”. Before entering the standby mode the SIUL_SCR0 bit shall be programmed to "0".

11.6.5 PDC

The PDC block is targeted to measure the level of a set of 14 pins and to generate a system wake-up event when the level is acknowledged at a specified level and, optionally, for a certain time.

Two commands are needed for the PDC:

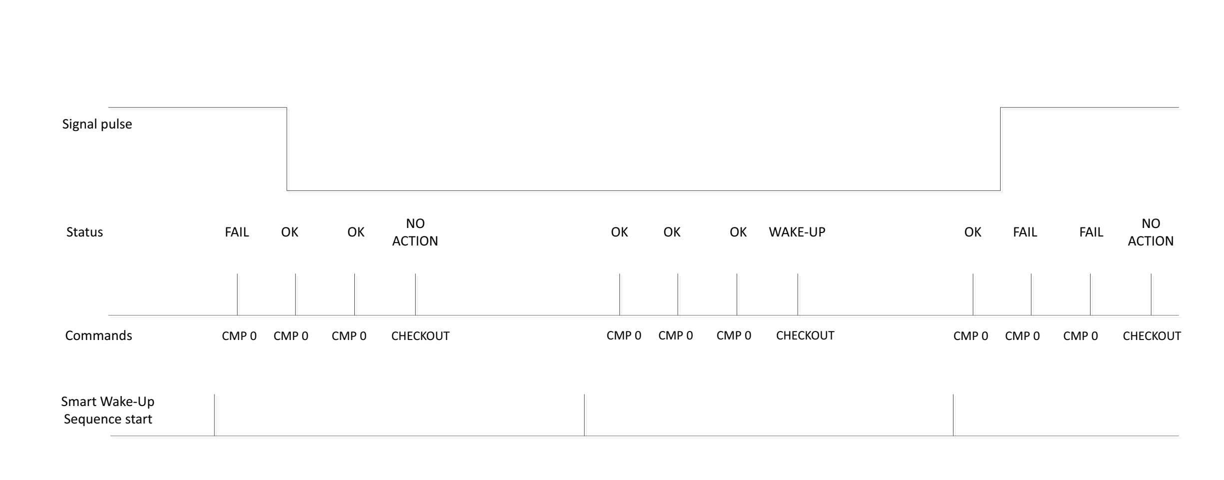
- CMP 0/CMP 1: to compare a pin level versus a 0 or 1
- CHECKOUT: to trigger a wake-up in case the result of the CMP commands for that specific pin executed from the beginning of the Smart Wake-up Sequence is always OK

For each pin, the PDC will latch the comparison results, which will report whether the target value was always matched (whenever a PDC, CMP0/1 command is launched) or not.

When checkout command is launched for a specific pin, it will trigger a wake-up event only if the target level was steadily acknowledged along all the comparisons done for that pin.

Figure 80 summarizes the above description.

Figure 80. Smart Wake-up Sequence PDC



11.6.6 TU

The TU block is used to create a programmable delay between the command execution. It needs only one command with the delay value programmed as a parameter.

The delay parameter is intended as number of IRCOSC periods (so that when TU = 1, then the delay is 1 clock cycle; TU = 2, then the delay is 2 clock cycles; and so on). The clock cycle can be adjusted with a linear prescaler, which is programmed through SSWU_CTRL_REG register of PMC_Dig module during the RUN mode, and whose value is preserved through the stand-by mode. The prescaler has a division factor from 1 to 128 (7 bits). Combining the TU parameter and the prescaler, the delay ranges from 62.5 ns (Clock period = 62.5 ns, PARAMS[6:0] = 1, prescaler = 1) up to 1.016 ms (Clock period = 62.5 ns, DELAY = 127, TU_{PRESC} = 128).

Note: TU_{PRESC} is defined in the SSWU_CTRL_REG. Please refer to Chapter 54: Power management controller digital interface (PMC_Dig).

Caution: The two commands which can cause wakeup (ADC-ADC or ADC-PDC or PDC-PDC) shall not be executed one after another without inserting a TU delay command.

11.6.6.1 SSWU STOP Command

In case the TU parameter (delay time) is set to 0, this is decoded as a command request to stop the whole SSWU sequence.

11.7 SSWU Configuration example

11.7.1 Description

In this scenario, the device stays in STANDBY and, every 32 msec, starts following the SSWU operations sequence:

- suddenly set OPC1 pin to '1'
- after 200 µsec, set OPC1 pin to '0'
- perform an ADC conversion (configuring an ADC threshold to wakeup the device)
- set a TU command for stopping the SSWU sequence

Device wakes-up from STANDBY only in case of ADC threshold overcoming.

11.7.2 Registers configuration

Following a pseudo-code that explains how to configure all the SSWU registers and all others IPs that are interconnected with it in order to get the overall STANDBY scenario

11.7.2.1 SSWU registers Configuration

```
//First command list
SSTDBY_CTU_0.CLR[0].R = 0x60001001; //set OPC1(PB9) pins to '1'

//Second command list
STDBY_CTU_0.CLR[2].R = 0x20000001;   //set OPC1(PB9) pins to '0'

STDBY_CTU_0.CLR[3].R = 0x20000C00; //convert internal channel  (AN82/PB10)

STDBY_CTU_0.CLR[4].R = 0x60000003; //add a delay of 0usec to stop the
sequence

STDBY_CTU_0.CLCR1.B.T0_INDEX = 0;   // trigger0 points to CLRN[0]
STDBY_CTU_0.CLCR1.B.T1_INDEX = 2;   // trigger0 points to CLRN[2]

STDBY_CTU_0.THCR1.R = 0x00001111;   // trigger0-1 enable
STDBY_CTU_0.TCRN[0].R = 0x00000001; // trigger0 generated quite immediately
STDBY_CTU_0.TCRN[1].R = 0x00000C80; // trigger1 generated 200us after the
API event

STDBY_CTU_0.TGSISR.R = 1;  // ctu_trg_in[0] (API) Rising edge Enable

STDBY_CTU_0.CTUCR.B.TGSISR_RE = 1;   // TGS Input Selection Register Reload
Enable

STDBY_CTU_0.CTUCR.B.GRE = 1;     // General Reload Enable
STDBY_CTU_0.CTUCR.B.CTU_FSM_reset = 1; // reset CTU FSM before
                                       // next stand-by entry
```

11.7.2.2 Other IP registers configuration

Following IPs should also be configured in order to complete the entire STANDBY scenario:

```
---------------------------------------------
Clock and Mode Entry configuration
---------------------------------------------

MC_ME.ME.R = 0x0000FFFF;         // Enable all modes

// Set RUN Configuration Registers
MC_ME.RUN_PCN[1].R=0x000000FE;       // Peripheral ON in every running mode
MC_ME.LP_PCN[1].R=0x00002500;

MC_ME.PCTL15.B.RUN_CFG=1;        // SIUL2
MC_ME.PCTL114.B.RUN_CFG=1;       // ADC_STDBY
MC_ME.PCTL242.B.RUN_CFG=1;       // STDBY_CTU_0

MC_CGM.AC0_DC2.R = 0x80000000;   // enable ADC_CLK
MC_CGM.AC0_DC4.R = 0x80000000;   // enable ECTU_CLK

//Enter DRUN to act the previous configurations
MC_ME.MCTL.R = 0x30005AF0;       // DRUN Mode & Key
MC_ME.MCTL.R = 0x3000A50F;       // DRUN Mode & Key

while(MC_ME.GS.B.S_MTRANS);      // Waiting for end of transaction
while(MC_ME.GS.B.S_CURRENT_MODE != 0x3);   // ME_GS Check DRUN mode has
successfully been entered

---------------------------
WKPU configuration
---------------------------

//Configure wakeup line 0 (SSWU wakeup event)
WKPU.WRER.R = 1;                 // Wakeup Request Enabled
WKPU.WIREER.R = 1;               // Rising Edge Event Enabled

---------------------------------------------
SAR_ADC_10bit_STDBY configuration
---------------------------------------------

SAR_ADC_10BIT_STDBY.MCR.B.OWREN = 1; // SAR ADC 0 overwrite enable set
SAR_ADC_10BIT_STDBY.MCR.B.MODE = 0; // Normal Mode
SAR_ADC_10BIT_STDBY.ICNCMR2.B.NCE_CH82 = 1; //sets channels 82 to be
converted in SARADC_10bit_standby (see table in ADC Config)
SAR_ADC_10BIT_STDBY.MCR.B.CTU_MODE = 1; // Trigger mode is selected
SAR_ADC_10BIT_STDBY.MCR.B.CTUEN = 1;    // CTU enabled
```

```
SAR_ADC_10BIT_STDBY.MCR.B.PWDN = 1; // SAR ADC 0 automatic power
                                   // up by SSWU during stand-by

//ADC_STANDBY_CH0 = Internal Channel 82:
SAR_ADC_10BIT_STDBY.WTHRHLRN[0].B.THRH = 0x333; // select high THRHLR0
treshold = 4V
SAR_ADC_10BIT_STDBY.ICWSELR11.B.WSEL_CH82 = 0;  // select THRHLR0 treshold
for CH88
SAR_ADC_10BIT_STDBY.ICWENR2.B.WEN_CH82 = 1;  // enable watchdog for CH82
SAR_ADC_10BIT_STDBY.WTIMR.B.MSKWDG0H = 1;   // enable high THRHLR0 treshold
interrupt

---------------------------
STANDBY configuration
---------------------------

PMCDIG.MISC_CTRL_REG.B.LPRCREG_ENB = 0;    // Enable LPRC in running modes

---------------------------
API/RTC configuration
---------------------------

RTC_API.RTCC.R = 0x80009001;    // Counter Enabled, API Enabled, LPRC
prescaled by 8 @128KHz used, Trigger enabled
RTC_API.APIVAL.R = 0x00001000;  // API period =32.25msec
```

Caution: For SSWU, ADCSAR must be used in CTU_TRIGGER mode.

11.7.2.3 Activate SSWU sequence

Once all previous register configurations have been configured, it just needs to perform STANDBY Mode entering, in order to get the desired scenario.

```
//Mode Entry Module (MC_ME): Enter STANDBY Mode
MC_ME.ME_STANDBY0_MC.IRCON.B = 0; /* Switch off IRCOSC during standby */
MC_ME.MCTL.R = 0xD0005AF0;       /*! Write Mode and Key | MC_ME.MCTL */
MC_ME.MCTL.R = 0xD000A50F;  /*! Write Mode and Key inverted | MC_ME.MCTL */
```

12 Security

All SPC58 family devices have a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access. Some security features are available in all device configurations while other more advanced security features are not available in all device configurations. These more advanced features require more than simply being turned on or off; they require significant involvement for implementation. Consequently, they provide an important opportunity to define the security level of the device.

12.1 Basic security

All SPC584Bx devices have the following basic set of security features.

- Device security feature based on the life cycle model with code and data access progressively more restricted as device matures through defined life cycle steps
- Memory security features:
  - NVM censorship support, password protection, one-time-programmable (OTP) flash memory areas, Flash erase counter and tamper detection
  - SRAM and caches initialized to a constant value after reset (power-on reset, when in a test mode, when not booting from internal flash memory); MBIST functionality used in case of power-on and destructive resets to initialize RAMs selected via configuration in UTEST flash; BAF software routines to initialize RAMs when not booting from internal flash memory
- Monitoring of operating conditions
- Unique ID for each device: each SPC584Bx device has a unique identification number stored in an OTP flash memory area which can be read by any of the cores on the device
- Secure watchdog timer
- Debugger access control

12.2 Advanced security

Depending on the device configuration, the following features are available:

- Customer programmable Hardware Security Module (HSM): a dedicated security subsystem that includes a processor core, dedicated SRAM, an encryption module, and exclusive access to secure areas of device flash memory; HSM runs code independently from the main device processor cores and can be used to implement advanced security and monitoring functions
- Advanced debugger access control
- Boot modes:
  - Trusted/secure boot support
  - Handshake with BAF supported

12.3 Detailed security information

Details of most of the SPC584Bx security features are published in a separate *SPC584Bx Microcontroller Security Reference Manual*, which is only available to qualified customers.

13 Debug and Trace

This chapter discusses the device-specific information for the debug modules. This device implements a high-speed serial Nexus trace auxiliary port. In addition, the LFAST module, a high speed calibration interface, is supported for run control and memory access. The LFAST signals are alternate functions on the device's JTAG pins. The device starts up in JTAG mode and the tool can request that the JTAG interface be converted to the LFAST high-speed interface. All standards used have freely available specifications for tool developers.

13.1 Core debug support

Internal debug support in the e200z420n3 cores allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. The cores have the capability of exiting reset halted in debug mode, such that no code runs. This allows debuggers to perform operations such as edit of overlay memory prior to user code execution.

For details, refer to *Chapter 57: e200z420n3 Core Debug Support*.

13.2 Run control and memory access

Run control is any operation required to control device operation. Run control includes starting and stopping the processor's execution of code, as well as accessing memory while the device is stopped to download code. The Debug and Calibration Interface (DCI) provides the central run control including cross-triggering of breakpoint conditions. This DCI provides support for the IEEE 1149.1 JTAG standard with the JTAG Controller (JTAGC) as well as the IEEE 1149.7 standard with the Compact JTAG (CJTAG) module. The JTAG interface supports both Boundary Scan and Debug modes. It is based on the IEEE 1149.1 and IEEE 1149.7 standards. The JTAG Controller (JTAGC) operates in the standard IEEE 1149.1 5-wire interface. The JTAGC can either be fed directly from the JTAG pins or the CJTAG module, which supports the IEEE 1149.7 standard.

The Sequence Processing Unit (SPU) provides cross-triggering of events from the device. The SPU can access some of the DCI run control features since it can control the $\overline{\text{EVTO}}$ pins and monitor the $\overline{\text{EVTI}}$ pins. Furthermore, the SPU can use system triggers to cause many different types of actions, including triggering a breakpoint cross-trigger (managed by the DCI).

Table 140. Reference links to related information

Related module	Reference
Production Device Debug and Calibration Interface (DCI)	*Chapter 58: Debug and Calibration Interface (DCI)*
JTAG Controller (JTAGC)	*Chapter 59: JTAG Controller (JTAGC)*
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)	*Chapter 60: IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)*

Table 140. Reference links to related information (continued)

Related module	Reference
JTAG Data Communication (JDC)	*Chapter 61: JTAG Data Communication (JDC)*
Sequence Processing Unit (SPU)	*Chapter 62: Sequence Processing Unit (SPU)*

13.2.1 Debug and Calibration Interface (DCI)

The Debug and Calibration Interface (DCI) module provides debug features. The DCI module includes the device JTAG Controller and the IEEE 1149.7 interface. The DCI provides the following features:

- Debug mode enable control for connected tools or software
- Port-sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- IEEE 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all the CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Simultaneous operation for debug features

The DCI sends and receives the $\overline{\text{EVTI}}$[1:0] and $\overline{\text{EVTO}}$[1:0] signals to/from the cores and other modules. *Table 141* summarizes the sources and destinations of $\overline{\text{EVTI}}$, $\overline{\text{EVTO}}$, and related signals for the device. For example, $\overline{\text{EVTI}}$[0] are outputs of the JTAGM and SPU, and inputs to the Cores, HSM and NXMC.

Table 141. DCI signals

Signal	Cores	HSM	NXMC	NPC	JTAGM	SPU	DTS
$\overline{\text{EVTI}}$[0]	I	I	I	—	O	I/O	—
$\overline{\text{EVTI}}$[1]	—	—	—	—	O	I/O	—
$\overline{\text{EVTO}}$[0]	O	O	O	—	I	I/O	—
$\overline{\text{EVTO}}$[1]	—	—	—	—	I	I/O	O
Core debug event	I	—	—	—	—	—	—
DCI cross triggering	O	—	—	—	—	—	—

The DCI sends the system debug signal, ipg_debug, to the following modules. For details regarding module functionality during system debug, refer to the individual module chapter.

- CAN
- DSPI
- eDMA
- eMIOS
- FCCU
- I2C
- LINFLEX
- NXMC
- PIT
- SARADC
- SSCM
- STM
- SWT

13.2.2 JTAG Controller (JTAGC)

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format. All Nexus debug registers are accessed and configured via the JTAG Controller.

13.2.3 Compact JTAG (CJTAG)

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities.

13.2.4 JTAG Data Communication (JDC)

The JTAG Data Communication (JDC) module allows both JTAG and software access to two registers. These two registers are the challenge/response password authorization registers used by the Hardware Security Module (HSM).

13.2.5 Sequence Processing Unit (SPU)

The Sequence Processing Unit (SPU) provides on-device logic analyzer trigger functions, such as performance monitoring, by using internal device signals as trigger sources. Performance monitor functions are available at both the system and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System-level performance monitor functions are integrated into the SPU complex trigger logic, thus allowing the counting and timing of any debug trigger combinations supported by the SPU.

Various clients generate watchpoints and other triggers when operating in Debug mode. The SPU collects these triggers and uses them as conditions for programmable sequences of states and resultant actions. The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, it achieves complex logic-analyzer-like behavior, providing vital real-time visibility and the ability to debug system activities.

13.2.5.1 SPU Level1 Input Mux Configuration registers

The SPU Level1 Mux inputs for the L1SELn registers (where n = 0–7) are described in the following tables. The L1SEL0 register fields are described in the following table.

Table 142. L1SEL0 register field descriptions

Field	Description
MUX 0	000 Reserved 001 Reserved 010 CPU2 instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 1	000 Reserved 001 Reserved 010 CPU2 instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 2	000 Reserved 001 Reserved 010 CPU2 instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event input 1 (watchpoint 10) 111 CPU2 debug event output 2 (watchpoint 20)
MUX 3	000 Reserved 001 Reserved 010 CPU2 instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 4	000 Reserved 001 Reserved 010 CPU2 instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 142. L1SEL0 register field descriptions (continued)

Field	Description
MUX 5	000 Reserved 001 Reserved 010 CPU2 instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 6	000 Reserved 001 Reserved 010 CPU2 instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event input 2 (watchpoint 11) 111 CPU2 debug event output 3 (watchpoint 21)
MUX 7	000 Reserved 001 Reserved 010 CPU2 instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL1 register fields are described in the following table.

Table 143. L1SEL1 register field descriptions

Field	Description
MUX 8	000 Reserved 001 Reserved 010 CPU2 instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 9	000 Reserved 001 Reserved 010 CPU2 instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 143. L1SEL1 register field descriptions (continued)

Field	Description
MUX 10	000 Reserved 001 Reserved 010 CPU2 instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event output 0 (watchpoint 12) 111 Reserved
MUX 11	000 Reserved 001 Reserved 010 CPU2 instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 12	000 Reserved 001 Reserved 010 CPU2 instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 13	000 Reserved 001 Reserved 010 CPU2 instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 14	000 Reserved 001 Reserved 010 CPU2 instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event output 1 (watchpoint 13) 111 Reserved
MUX 15	000 Reserved 001 Reserved 010 CPU2 instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL2 register fields are described in the following table.

Table 144. L1SEL2 register field descriptions

Field	Description
MUX16	000 Reserved 001 Reserved 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 17	000 Reserved 001 Reserved 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 18	000 Reserved 001 Reserved 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 19	000 Reserved 001 Reserved 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 20	000 Reserved 001 Reserved 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to L1CSR0 111 Reserved
MUX 21	000 Reserved 001 Reserved 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to L1CSR1 111 Reserved

Table 144. L1SEL2 register field descriptions (continued)

Field	Description
MUX 22	000 Reserved 001 Reserved 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to E2ECTL0 111 Reserved
MUX 23	000 Reserved 001 Reserved 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to DMEMCTL0 111 Reserved

The L1SEL3 register fields are described in the following table.

Table 145. L1SEL3 register field descriptions

Field	Description
MUX 24	000 Reserved 001 Reserved 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to IMEMCTL0 111 Reserved
MUX 25	000 Reserved 001 Reserved 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to HID0 111 Reserved
MUX 26	000 Reserved 001 Reserved 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to HID1 111 Reserved

Table 145. L1SEL3 register field descriptions (continued)

Field	Description
MUX 27	000 Reserved 001 Reserved 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to BUCSR 111 Reserved
MUX 28	000 Reserved 001 Reserved 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to SPEFSCR 111 Reserved
MUX 29	000 Reserved 001 Reserved 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to IVPR 111 Reserved
MUX 30	000 Reserved 001 Reserved 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to MSR 111 Reserved
MUX 31	000 Reserved 001 Reserved 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL4 register fields are described in the following table.

Table 146. L1SEL4 register field descriptions

Field	Description
MUX 32	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Device NPC full 111 Reserved
MUX 33	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Device NPC partial full 111 Reserved
MUX 34	000 Reserved 001 Reserved 010 CPU2 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 35	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 36	000 CPU2 performance monitor counter 0 (watchpoint 23) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 37	000 CPU2 performance monitor counter 1 (watchpoint 24) 001 Reserved 010 Reserved 011 Reserved 100 NXMC1 watchpoint 7 101 Reserved 110 Reserved 111 Reserved

Table 146. L1SEL4 register field descriptions (continued)

Field	Description
MUX 38	000 CPU2 performance monitor counter 2 (watchpoint 25) 001 Reserved 010 CPU2 performance monitor event input (watchpoint 22) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 39	000 CPU2 performance monitor counter 3 (watchpoint 26) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL5 register fields are described in the following table.

Table 147. L1SEL5 register field descriptions

Field	Description
MUX 40	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 0 111 Reserved
MUX 41	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 1 111 Reserved
MUX 42	000 CPU2 interrupt (watchpoint 16) 001 CPU2 critical interrupt (watchpoint 18) 010 CPU2 data trace control range 1 (watchpoint 29) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 2 111 Reserved

Table 147. L1SEL5 register field descriptions (continued)

Field	Description
MUX43	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 3 111 Reserved
MUX 44	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 4 111 Reserved
MUX 45	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 5 111 Reserved
MUX 46	000 CPU2 return (watchpoint 17) 001 CPU2 critical return (watchpoint 19) 010 CPU2 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 6 111 Reserved
MUX 47	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 7 111 Reserved

The L1SEL6 register fields are described in the following table.

Table 148. L1SEL6 register field descriptions

Field	Description
MUX 48	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 0 100 Reserved 101 Reserved 110 SPU counter event 8 111 Reserved
MUX 49	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 1 100 Reserved 101 Reserved 110 SPU counter event 9 111 Reserved
MUX 50	000 CPU2 context load (using e_lmvsprw instruction) 001 CPU2 context restore (using e_stmvsprw instruction) 010 CPU2 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 2 100 Reserved 101 Reserved 110 SPU counter event 10 111 Reserved
MUX 51	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 3 100 Reserved 101 Reserved 110 SPU counter event 11 111 Reserved
MUX 52	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 4 100 Reserved 101 Reserved 110 SPU counter event 12 111 Reserved
MUX 53	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 5 100 Reserved 101 Reserved 110 SPU counter event 13 111 Reserved

Table 148. L1SEL6 register field descriptions (continued)

Field	Description
MUX 54	000 CPU2 interrupt current priority match to C2PIS 001 CPU2 write to ME bits of MSR 010 Reserved 011 NXMC0 watchpoint 6 100 Reserved 101 Reserved 110 SPU counter event 14 111 Reserved
MUX 55	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 7 100 Reserved 101 Reserved 110 SPU counter event 15 111 Reserved

The L1SEL7 register fields are described in the following table.

Table 149. L1SEL7 register field descriptions

Field	Description
MUX 56	000 Reserved 001 Reserved 010 NXMC1 watchpoint 0 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTI0
MUX 57	000 Reserved 001 Reserved 010 NXMC1 watchpoint 1 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTI1
MUX 58	000 CPU2 write to EE bits of MSR 001 CPU2 write to CE bits of MSR 010 NXMC1 watchpoint2 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO0

Table 149. L1SEL7 register field descriptions (continued)

Field	Description
MUX 59	000 Reserved 001 Reserved 010 NXMC1 watchpoint 3 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO1
MUX 60	000 Reserved 001 Reserved 010 NXMC1 watchpoint 4 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI system halt
MUX 61	000 Reserved 001 Reserved 010 NXMC1 watchpoint 5 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 62	000 CPU2 processor exception vector match to C2PEVP 001 CPU2 write to DE bits of MSR 010 NXMC1 watchpoint 6 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 63	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

13.2.5.2 Tool interface

This device includes an alternate calibration interface for high-speed run control and advanced software control. This calibration interface is based on the LFAST interface. The LFAST based calibration has the ability to perform all of the functions that JTAG supports, only at a faster speed. The MCU implements the slave module, and the pins for the LFAST interface are multiplexed onto the same pins as the JTAG pins. Initially, the device powers up with the JTAG module in control of the pins. Under tool commands, the LFAST interface can be enabled and the pins become LFAST signals. Once the LFAST interface has been activated, all run control and other JTAG operations are converted in the MCU from LFAST messages into JTAG control operations of the JTAG Controller. The interface is supported entirely in hardware for both LFAST and JTAG modes, with no software or system overhead required to support interface traffic.

Table 150. Reference links to related information

Related module	Reference
JTAG Master (JTAGM)	*Chapter 63: JTAG Master (JTAGM)*
LFAST Module (Debug LFAST)	*Chapter 66: LVDS Fast Asynchronous Serial Transmission – High Speed Debug*
Development Tool Semaphore (DTS)	*Chapter 67: Development Trigger Semaphore (DTS)*

13.2.6 JTAG Master (JTAGM)

The JTAG Master (JTAGM) acts as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or through software.

The data transferred to this module is transformed to produce TCK, TMS, TDI, and TRST outputs and to accept TDO inputs. The module allows these five signals to connect to the JTAGC as if the JTAG data were coming from an external tool. The JTAGM generates all required JTAG scan chains to allow software and high-speed serial communication access to all JTAG-mapped resources.

13.2.7 Debug LFAST

The high-speed calibration interface provides up to a 320 Mbit/s interface between a calibration tool and the device. The calibration interface uses the LFAST as the physical layer between the tool and the JTAGM for access to debug resources. In addition, an LFAST switch allows direct access to read and write memory via the SIPI_DEBUG module. The calibration LFAST is configured for slave-only operation and is meant to be used as a calibration interface to provide higher (than JTAG) speed debug and calibration operations.The calibration LFAST interface supports full duplex operation, where the available bandwidths for upstream and downstream traffic are independent.

13.2.8 Development Trigger Semaphore (DTS)

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There is a variety of ways that this module can be used, including as a component of an external real-time data acquisition system.

Note: When used as a component of a triggered data-acquisition system, Nexus read/write access is via the JTAG interface of the Nexus debug port and is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus auxiliary port.

13.3 Debug over CAN

As well as supporting debug via JTAG and LFAST interfaces, the device supports the use of a CAN interface for debug. This allows debug of application hardware where access to the dedicated JTAG/LFAST interface signals is not easily available. The debug over CAN mechanism is intended to provide basic debug functionality with some reduction in bandwidth and features compared to the use of the dedicated JTAG or LFAST interfaces.

Use of the CAN interface for debug purposes requires the use of some M_CAN, eDMA and JTAGM resources. Some initialization of these resources via software is required, but once this initialization has completed, debug over CAN is possible without any further software overhead.

As the debug over CAN scheme generates internal JTAG messages based on received CAN data, all JTAG clients and included debug resources are accessible. Basic device trace capability is also possible by configuring the device trace hardware to stream to a device overlay/trace RAM, which can be read later using debug over CAN.

The debug over CAN scheme supports the following features:

- Supports operation through CAN_SUB_0_M_CAN_1 or CAN_SUB_0_M_CAN_2 interfaces
  - Makes use of M_CAN debug enhancements
  - Debug message selected by filter configuration SFEC/EFEC=111
- Allows debug of hardware where JTAG access is not available
- Debug traffic carried over application CAN bus
- Debug traffic on CAN coexists with application traffic
- No software overhead after initialization
- flexible selection of CAN identifiers for debug use
- Access to all JTAG debug facilities, includes CPU run control

13.4 Nexus Trace interface

This interface allows Nexus protocol information to be transmitted at high speed.

The trace information can be used to reconstruct events or operations that occurred inside the microcontroller. Nexus supports the transmission, through a single port, of information from multiple trace clients within the MCU. Each Nexus message is tagged with the source client identifier and the type of message. The actual trace information that is available depends on the Nexus client; generally, it can be individually enabled (both the type of message and the client). *Table 151* shows a few examples of Nexus messages. All messages can be enabled to include a timestamp.

Table 151. Nexus message types

Message Type	Description
Program trace	Nexus program trace messages transmit any discontinuities in the program flow, such as branches and interrupts. Multiple types of messages can be generated.
Ownership trace	Ownership trace provides information on process identification changes.

Table 151. Nexus message types (continued)

Message Type	Description
Watchpoint trace	Watchpoint messages are generated any time a watchpoint match occurs in the program or data flow, or as a result of SPU actions. These can be programmed for many types of events within the MCU.
Device identification	The device identification message allows information about the MCU to be transmitted upon startup to allow tools to identify the target system MCU type.
Debug status	Debug status messages provide additional information that may be needed to reconstruct software execution, such as whether the device has entered Debug or Low-power mode.
Data acquisition	Data acquisition messages are an optional feature that allows software control of information to be transmitted.
error	Error messages are transmitted when an error condition occurs, such as internal buffer overflows.

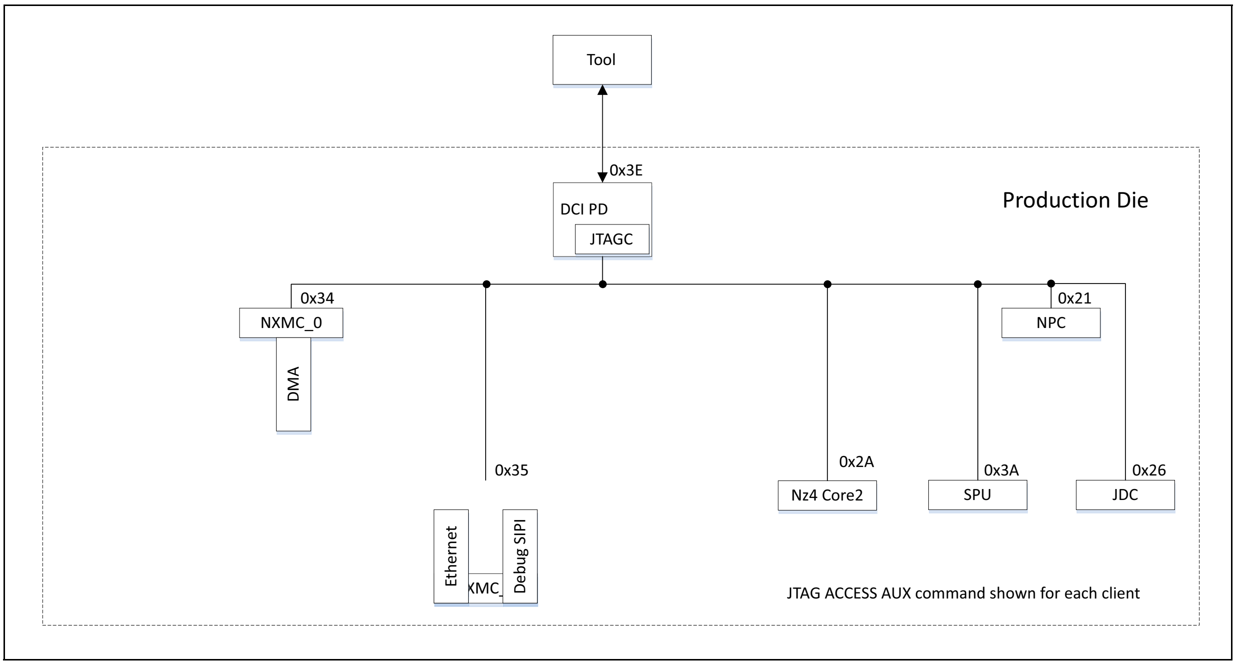
The Nexus blocks do not have memory-mapped registers. The Nexus registers are accessed by the development tool via the JTAG port using a two step process. First, the specific block is selected by loading the corresponding ACCESS_AUX instruction as described in the JTAGC chapter. Next, after the block is selected, the Nexus registers are enabled by loading the enable instruction to the JTAGC Instruction Register (IR). The enable instructions for the different Nexus clients are shown in the following table. After the enable instruction is received, the development tool has access to the Nexus registers of the selected client.

Table 152. Nexus client JTAG enable instructions

Module	Enable instruction	Opcode
Production Die		
Cores	CORE_ENABLE	0x7C (10 bits)
NXMC	NEXUS_ENABLE	0x0 (4 bits)
NPC		
SPU		

All of the internal JTAG modules are accessed through a single JTAG connection to the device.

Figure 81. JTAG hierarchical view



13.4.1 Nexus Port Controller (NPC)

The Nexus Port Controller controls the usage and sharing of the Nexus Auxiliary Port between Nexus clients. The Nexus Auxiliary Port is accessed by:

- The N3 block of the CPU
- The DMA master port NXMC client
- The Ethernet
- The SIPI_DEBUG

The NPC is placed in disabled mode upon exit of reset. Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. Once the NPC is enabled, if message transmission via the auxiliary port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

13.5 Nexus clients

Nexus supports the transmission of information from multiple trace clients within the device.

Table 153. Reference links to related information

Related module	Reference
e200z420n3 Nexus Support	*Chapter 70: e200z420n3 Nexus 3 Module*
Nexus Crossbar Multi-Master Client (NXMC)	*Chapter 71: Nexus Crossbar Multi-Master Client (NXMC)*

13.5.1 e200z420 Nexus 3

The e200z420n3 Nexus 3 module provides real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering and Processor Overrun Control.

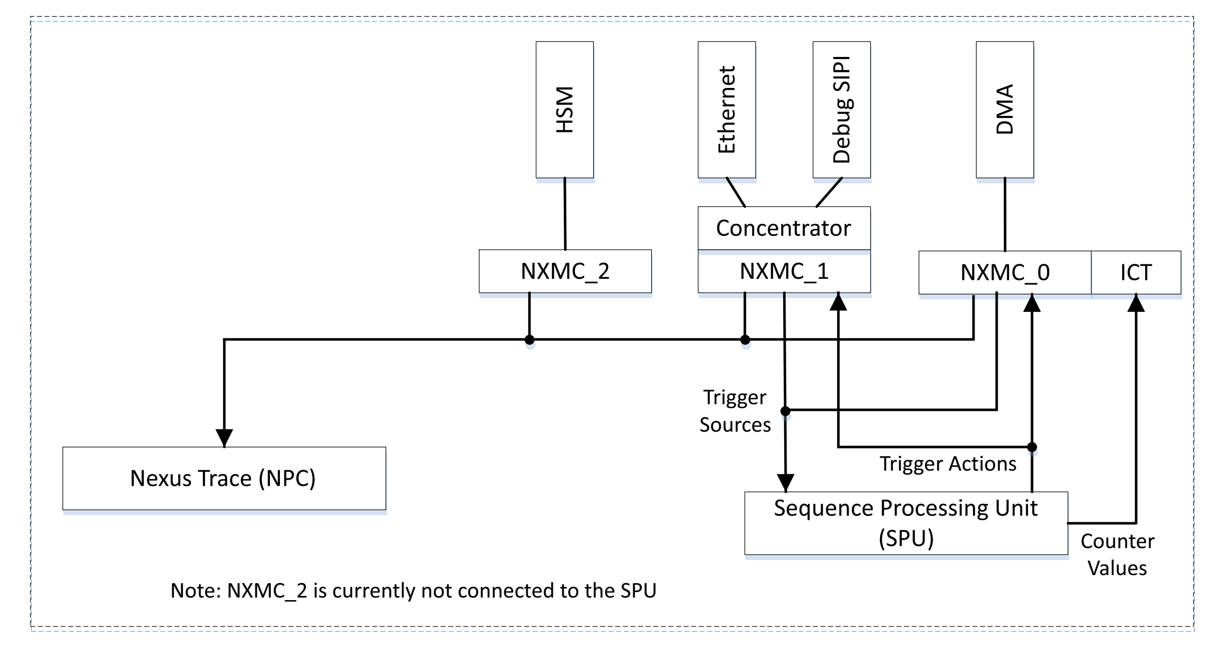
13.5.2 Nexus Crossbar Multi-master Client (NXMC)

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2011 standard. This module provides development support capabilities for SoCs without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard.

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Three NXMCs are implemented, out of which only two interact with the SPU. Trace messages transmitted over the Nexus interface include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message. In addition to XBAR bus master trace capability, the NXMC also allows for in-circuit trace capability that can be used to trace the Sequence Processing Unit (SPU) counter values. The two external hardware watchpoint triggers of each NXMC are used to generate timestamps (four in total: two from each NXMC) for events from the SPU.

The *Figure 82* shows a block diagram illustrating the NXMC integration.

Figure 82. NXMC integration block diagram



The values of the trace master ID or in-circuit client ID, along with the Nexus message source identifier for the different clients are shown in the following table.

Table 154. NXMC source and master client IDs

Client		Nexus trace source ID (SRC)	Trace master ID or In-circuit trace client ID (MASTER)
In-Circuit Trace	SPU	0b1011	0b0000
direct master	eDMA Controller	0b1100	0b0001
XBAR Client 1 Concentrator	Debug SIPI	0b1101	0b1000
	Ethernet_0		0b1010
direct master	HSM	0b1000	0b0000

14 e200z4d Core Complex Description

The SPC584Bx has one e200z420n3 core that performs various computational and control functions. This core is used for general computational functions.

This chapter provides overviews of the e200z420n3.

14.1 Overview of the e200z420n3 core

The e200z processor family is a set of CPU cores that implement low-cost versions of the Power architecture.

The e200z420n3 are dual-issue, 32-bit Power VLE compliant designs with 32-bit general purpose registers (GPRs).

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision embedded numeric operations using the general-purpose registers.

The e200z420n3 cores implement the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document. The base *PowerISA 2.06* fixed-length 32-bit instruction set is not directly supported.

The e200z420n3 processor cores integrate a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

A Nexus Class 3+ module is integrated into each e200z420n3 processor core.

A Memory Protection Unit is integrated in each e200z420n3 processor core.

14.2 Features

The following is a list of some of the key features of the e200z420n3 processor cores:

- Dual issue, 3[2-bit *PowerISA 2.06*-VLE compliant CPU
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - Dedicated branch address calculation adder
  - Branch target prefetching using BTB
  - Return Address Stack
- Load/store unit
  - 2 cycle load latency
  - Fully pipelined
  - Misaligned access support
- 32-bit General Purpose Register (GPR) file
- Dual AHB 2.v6 64-bit System buses
- Local Data Memories (DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z420n3 core: 64 KB DMEM
- Memory Protection Unit (MPU) implementing a 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized instruction or data memory regions
- 2-Way Set Associative Harvard Instruction and Data Cache
  - e200z420n3 core: 8 KB ICache, 4 KB DCache
- Embedded Floating-point APU (EFPU2) supporting single-precision floating-point operations
- Performance Monitor APU supporting execution profiling
- Nexus Class 3-plus Real-time Development Unit
- Power management
  - Low power design with extensive clock gating
  - Power saving modes: doze, nap, sleep, wait
  - Dynamic power management of execution units, caches and Local memories
- Testability
  - Synthesizeable, MuxD scan design
  - ABIST/MBIST for arrays

14.3 Microarchitecture summary

The e200z420n3 processor cores utilize a five stage instruction pipeline with two stages for execution. The Instruction Fetch, Instruction Decode/Register File Read/ EA Calc, Execute 0/ Memory Access0, Execute1/Memory Access1, and Register Writeback stages operate in an overlapped fashion allowing single clock instruction execution for most instructions. *Figure 83* shows a block diagram of the e200z architecture.

The integer execution units each consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), and result feed-forward hardware. The integer EU1 also supports hardware division and a 32x32 Hardware Multiplier array.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a pipelined hardware array, and the divide instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

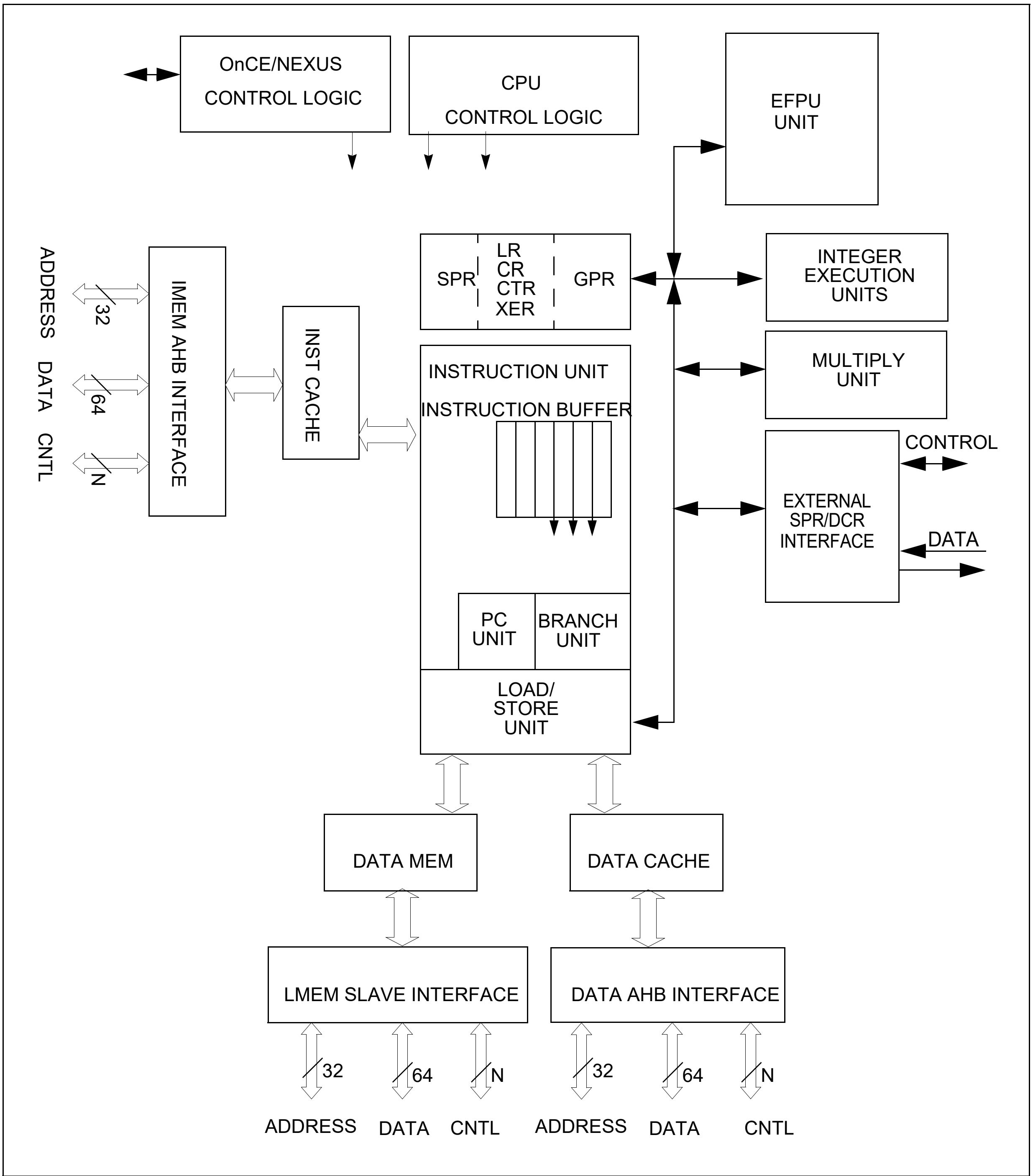
The Instruction Unit contains a PC incrementer and dedicated Branch Address adders to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in an execution time of two clocks for correctly predicted branches. Conditional branches which are not taken execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, word (32-bit), and doubleword data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

The Condition Register unit supports the condition register (CR) and condition register operations defined by the Power architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Figure 83. e200z block diagram



Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

14.3.1 Instruction unit features

The features of the e200z420n3 Instruction units are:

- 64-bit path to cache supports fetching of two 32-bit instructions per clock
- Instruction buffer holds up to 8 32-bit instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder and branch lookahead logic (BTB) supporting single cycle execution of successfully predicted branches

14.3.2 Integer unit features

The e200z420n3 integer units support single cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for the count leading zero's function
- 32-bit single cycle barrel shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 4-14 clocks with minimized execution timing (EU1 only)
- Pipelined 32x32 hardware multiplier array supports 32x32->32 multiply with 2 clock latency, 1 clock throughput (EU1 only)

14.3.3 Load/Store unit features

The e200z420n3 load/store units support load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions and context save/restore instructions

14.3.4 MPU features

The features of the MPU are as follows:

- 24-entry fully-associative Range Table
- Arbitrary range size support from 1 byte to 4 GB
- 8-bit Process Identifier
- Bypass capability for individual access types
- Maskable Address and Process Identifier
- Programmable Memory Attributes
- Entry Invalidation Protection
- Write-Once Option
- Alternate Debug Breakpoint/Watchpoint function

14.3.5 Cache features

The features of the Caches are as follows:

- 2-Way Set Associative Harvard Instruction and Data Caches
  - e200z420: 8 KB ICache, 4 KB DCache
- Write through Support
- 8-entry Store Buffer
- Linefill Buffer
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and 64-bit write data bus
- Supports Way locking
- Supports Write allocation policies
- Supports multi-bit EDC with Correction/Auto-invalidation capability for the I and D caches
- Hardware Debug Cache Invalidate Support for the Data Cache
- Software and Hardware Debug access to cache tag, status and data storage via dedicated control registers

14.3.6 Local memory features

The features of the Local Instruction and Data Memories are as follows:

- Data Memories (DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z420n3: 64 KB DMEM
- Supports zero wait-state access relative to cache hit timings
- Supports multi-bit (SECDED) ECC with correction/scrubbing capability
- 4-entry ECC R-M-W Store Buffer for DMEM
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and (for DMEM only) 64-bit write data bus

14.3.7 e200z420n3 system bus features

The features of the e200z420n3 System Bus interface are as follows:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture (AMBA) AHB2.v6 protocol
- 32-bit address bus, 64-bit data bus, plus attributes and control
- Data interface provides separate uni-directional 64-bit read and write data buses
- Support for HCLK running at a slower rate than CPU clock

15 Core (z420n3) Description

15.1 Overview of the Core (z420n3)

Core (z420n3) is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general purpose registers (GPRs). The Core (z420n3) core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

The Core (z420n3) processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

Core (z420n3) contains an 8 KB Instruction Cache, and a 4 Kbyte Data Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit is also included which supports protections of various instruction and data memory areas.

15.2 Register Model

This section describes the registers implemented in the Core (z420n3) core. The *PowerISA 2.06* architecture defines register-to-register operations for all computational instructions.

Core (z420n3) extends usage of the General Purpose Registers to support EFPU APU operations operating on the 32-bit GPR registers.

Figure 84 and *Figure 85* show the complete Core (z420n3) register set, all of which are accessible in supervisor mode. *Figure 86* and *Figure 87* show the subset of registers which are accessible while in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

Figure 84. Core (z420n3) Supervisor Mode Programmer's Model SPRs

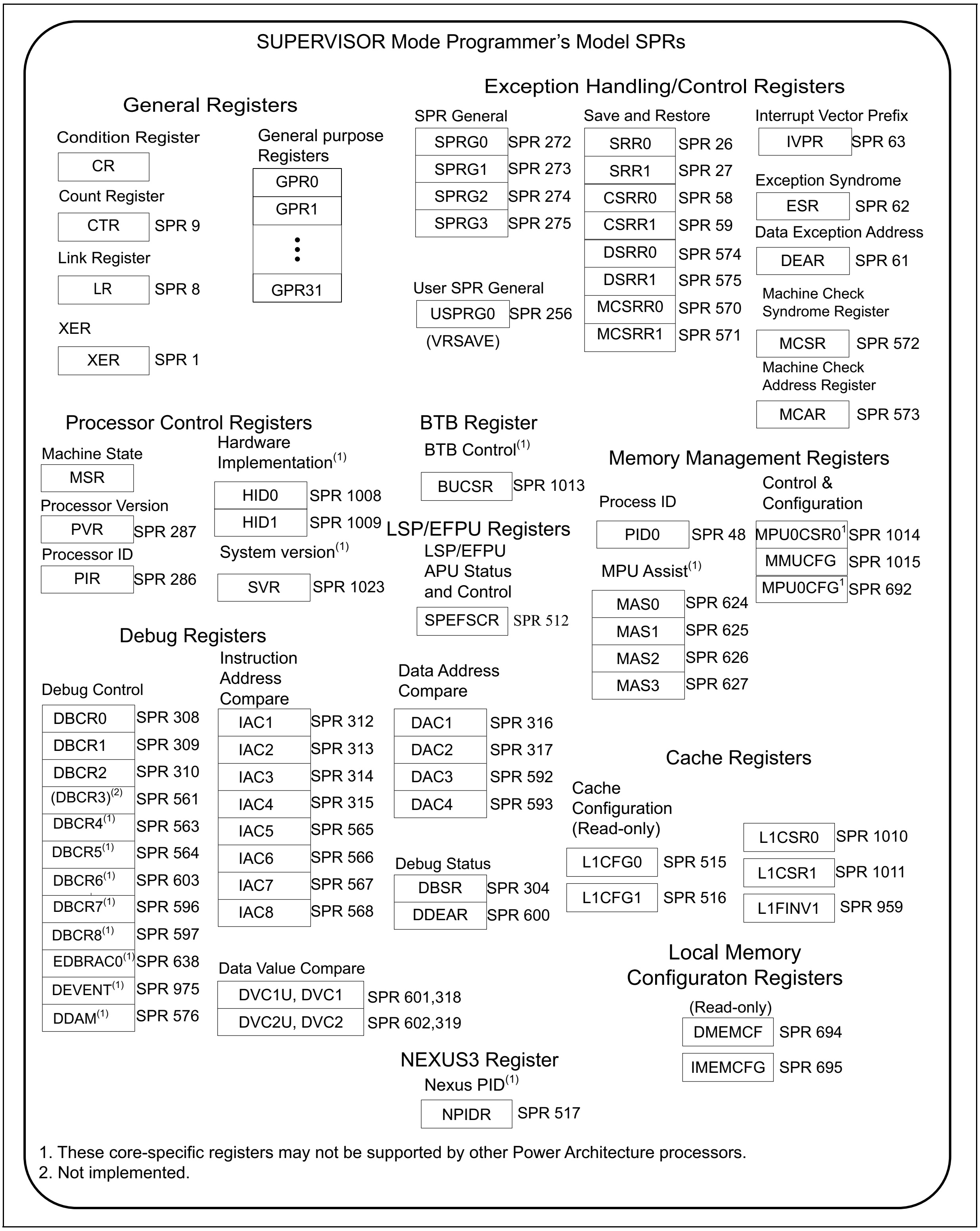


Figure 85. Core (z420n3)Supervisor Mode Programmer's Model DCRs and PMRs

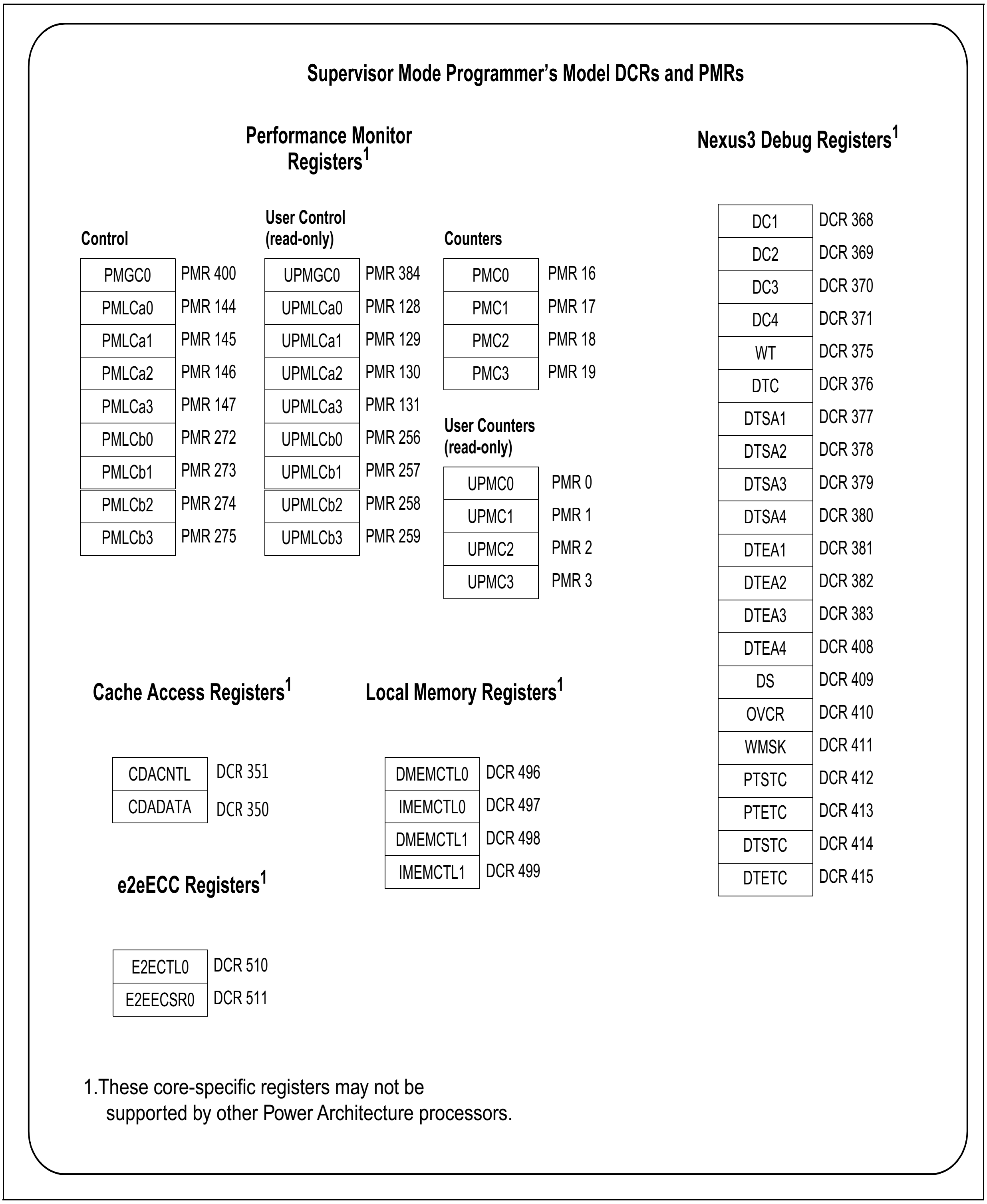


Figure 86. Core (z420n3) User Mode Programmer’s Model SPRs

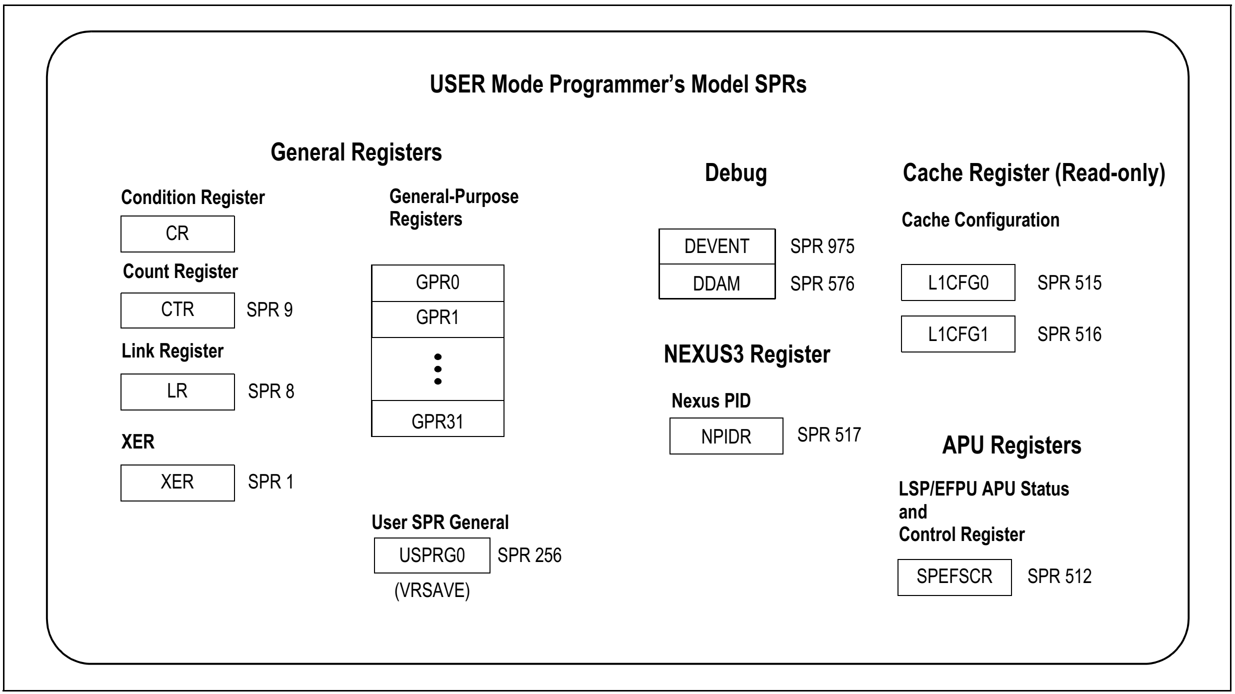
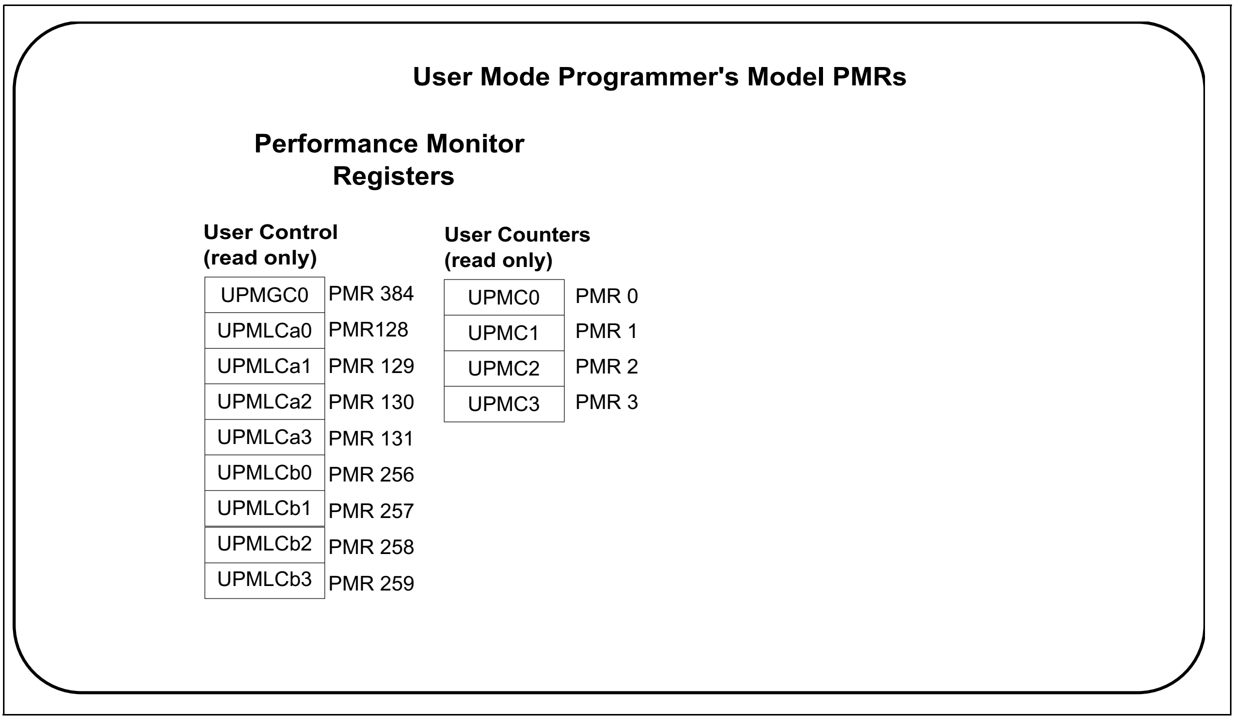


Figure 87. Core (z420n3) User Mode Programmer’s Model PMRs



15.3 Dual Issue Operation

The instruction issue unit attempts to issue a pair of instructions each cycle to the execution units. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions which cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

Table 155 shows the Core (z420n3) concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue.

Table 155. Concurrent Instruction Issue Capabilities

Class of Instruction	branch	load/store	scalar integer	scalar float	vector integer non-MAC	vector integer MAC	special
branch	—	yes	yes	yes	yes	yes	—
load/store	yes	—	yes	yes	yes	yes	—
scalar integer	yes	yes	yes(1)	yes	yes	yes(2)	—
scalar float	yes	yes	yes	—	yes	yes	—
vector integer non-MAC	yes	yes	yes	yes	—	—	—
vector integer MAC	yes	yes	yes(2)	yes	—	—	—
special	—	—	—	—	—	—	—

1. excludes multiply or divide class instructions occurring in both issue slots
2. excludes vector MAC/multiply class instructions occurring with scalar multiply, or divide class instructions

15.4 Reservation Instructions and Cache Interactions

The Zen core treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx.**, **sthcx.**, and **stwcx.**) accesses as though they were cache inhibited, and forces a cache miss. A reservation access is always issued to the bus.

15.5 Signal Processing Extension / Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point uses the SPEFSCR register. The SPEFSCR register is implemented as special purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in *Figure 88*.

0	FINXS	FINVS	FDBZS	FUNFS	FOVFS	0	SOV	OV	FG	FX	FINV	FDBZ	FUNF	FOVF	0	FINXE	FINVE	FDBZE	FUNFE	FOVFE	FRMC
0 1 2 3 4 5 6 7 8 9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30 31

SPR - 512; Read/Write; Reset - 0x0

Figure 88. DSP/EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in *Table 156*.

Table 156. EFPU Status and Control Register

Bits	Name	Description
0:9	—	Reserved
10	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF=1), but Floating-point Overflow exceptions are disabled (FOVFE=0), or if the result of a Floating-point instruction results in underflow (FUNF=1), but Floating-point Underflow exceptions are disabled (FUNFE=0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
11	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
12	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
13	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
14	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
15	—	Reserved
16	SOV	Summary integer overflow Defined by LSP APU.
17	OV	Integer overflow Defined by LSP APU.
18	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler.FX is zeroed if a Floating-point Data Exception occurs.

Table 156. EFPU Status and Control Register (continued)

Bits	Name	Description
20	FINV	Embedded Floating-point Invalid Operation / Input error The FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0.
21	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24	—	Reserved
25	FINXE	Embedded Floating-point Inexact Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF=1) but Floating-point Overflow exceptions are disabled (FOVFE=0), or if the result of a Floating-point instruction results in underflow (FUNF=1 or FUNFH=1) but Floating-point Underflow exceptions are disabled (FUNFE=0), and no Floating-point Data exception occurs.
26	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction.
27	FDBZE	Embedded Floating-point Divide by Zero Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction.
28	FUNFE	Embedded Floating-point Underflow Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction.

Table 156. EFPU Status and Control Register (continued)

Bits	Name	Description
29	FOVFE	Embedded Floating-point Overflow Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction.
30:31	FRMC	Embedded Floating-point Rounding Mode Control 00 = Round to Nearest 01 = Round toward Zero 10 = Round toward +Infinity 11 = Round toward -Infinity

15.6 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

15.6.1 Cache Overview

The Core (z420n3) processor supports a 8 Kbyte 2-way set-associative instruction and 4 Kbyte 2-way set associative data cache with a 32-byte line size. The caches improve system performance by providing low-latency data to the Core (z420n3) instruction and data pipelines, which decouples processor performance from system memory performance.

Instruction and data addresses from the processor are used to index the cache array. If the access address matches a valid cache tag entry, the access hits in the cache.

15.6.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache as well as providing general control over disabling ways in both caches. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR0 is 1010 in decimal. The L1CSR0 register is shown in *Figure 89*.

WID	0	WDD	0	0	DCWA	0	DCECE	DCEI	DCLOC	0	DCEA	DCLOINV	0	DCABT	DCINV	DCE
0 1	2 3	4 5	6	7 8 9 10 11	12	13 14	15	16	17 18	19 20 21 22 23 24	25 26	27	28	29	30	31

SPR - 1010; Read/Write; Reset - 0x0

Figure 89. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 bits are described in *Table 157*.

Table 157. L1CSR0 Field Descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable. 0 = The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 = The corresponding way instruction cache is not available for replacement by instruction miss line fills. Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache.
2:3	—	Reserved(1)
4:5	WDD	Way Data Disable. 0 = The corresponding way in the data cache is available for replacement by data miss line fills. 1 = The corresponding way in the data cache is not available for replacement by data miss line fills. Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache.
8:11	—	Reserved(1)
12	DCWA	Data Cache Write Allocation Policy 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled When enabled is a write misses the cache the cache line is loaded into cache before writing data.
13:14	—	Reserved(1)
15	DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	DCEI	Data Cache Error Injection 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits. This includes writes due to store hits as well as writes due to cache line refills. DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE=0.

Table 157. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
17:18	DCLOC	Data Cache Lockout Control 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error. **Note:** For the **dcbi** and **dcbf** instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of error conditions.
19:24	—	Reserved(1)
25:26	DCEA	Data Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 1x Reserved
27	DCLOINV	Data Cache Lockout Indicator Invalidate When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation When written to a ‘1’ in conjunction with writing DCINV to a ‘1’, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value.
28	—	Reserved(1)
29	DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

Table 157. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
30	DCINV	Data Cache Invalidate 0 No cache invalidate 1 Cache invalidation operation When written to a ‘1’, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value.
31	DCE	Data Cache Enable 0 Cache is disabled 1 Cache is enabled When disabled, cache lookups are not performed for normal load or store accesses. Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE.

1. These bits are not implemented and should be written with zero for future compatibility.

15.6.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR1 is 1011 in decimal. The L1CSR1 register is shown in *Figure 90*.

0–14	15	16	17–18	19–24	25–26	27	28	29	30	31
0	ICECE	ICEI	ICLOC	0	ICEA	ICLOINV	0	ICABT	ICINV	ICE

SPR - 1011; Read/Write; Reset - 0x0

Figure 90. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 bits are described in *Table 158*.

Table 158. L1CSR1 Field Descriptions

Bits	Name	Description
0:14	—	Reserved(1)
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits. ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE=0.

Table 158. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
17:18	ICLOC	Instruction Cache Lockout Control 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error. **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of error conditions.
19:24	—	Reserved(1)
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 1x Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation When written to a ‘1’ in conjunction with writing ICINV to a ‘1’, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value.
28	—	Reserved(1)
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

Table 158. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
30	ICINV	Instruction Cache Invalidate 0 No cache invalidate 1 Cache invalidation operation When written to a ‘1’, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value.
31	ICE	Instruction Cache Enable 0 Cache is disabled 1 Cache is enabled When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE.

1. These bits are not implemented and should be written with zero for future compatibility.

15.6.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the Core (z420n3) L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The SPR number for L1CFG0 is 515 in decimal. The L1CFG0 register is shown in *Figure 91*.

CARCH	CWPA	CFAHA	DCFISW	0		DCBSIZE	DCREPL	DCLA	DCECA	DCNWAY	DCSIZE
0 1	2	3	4	5	6	7 8	9 10	11	12	13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30 31
00	1	0	1	0	0	00	11	0	1	00000001 (2 way)	00000000100 (4 KB)

SPR - 515; Read-only

Figure 91. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 bits are described in *Table 159*.

Table 159. L1CFG0 Field Descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 - The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available 1 - The caches support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 - The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 - The data cache supports invalidation by Set and Way via L1FINV0

Table 159. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
5:6	—	Reserved - read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 - The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11- The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 - The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 - The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 - The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x004 - The size of the data cache is 4 Kbyte

15.6.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the Core (z420n3) L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The SPR number for L1CFG1 is 516 in decimal. The L1CFG1 register is shown in *Figure 92*.

0	ICFISW	0	ICBSIZE	ICREPL	ICLA	ICECA	ICNWAY	ICSIZE
0 1 2 3	4	5 6	7 8	9 10	11	12	13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30 31
0000	1	0 0	00	11	0	1	00000001 (2 way)	00000001000 (8 KB)

SPR - 516; Read-only

Figure 92. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 bits are described in *Table 160*.

Table 160. L1CFG1 Field Descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 - The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 - The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 - The instruction cache implements a FIFO replacement policy

Table 160. L1CFG1 Field Descriptions (continued)

Bits	Name	Description
11	ICLA	Instruction Cache Locking APU Available 0 - The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 - The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 - The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 - The size of the instruction cache is 8 Kbyte

15.6.6 Data Cache Software Coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

15.6.7 Data Cache Hardware Coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency. Debug support is provided however for a Dcache line invalidation operation requested by an external hardware debugger to occur. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available DCache access cycle.

15.6.8 Cache Invalidate by Set and Way

Core (z420n3) supports cache set/way invalidation under software control. The caches may be invalidated by set and way through a **mtspr l1finv{0,1}** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0,1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

15.6.8.1 L1FINV0

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in *Figure 93*.

Bits 0–6	Bit 7	Bits 8–20	Bits 21–26	Bits 27–29	Bits 30–31
0	CWAY	0	CSET	0	CCMD

SPR - 1016; Read/Write; Reset - 0x0

Figure 93. L1 Flush/Invalidate Register 0 (L1FINV0)

The L1FINV0 bits are described in *Table 161*.

Table 161. L1FINV0 Field Descriptions

Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way Specifies the data cache way to be selected
8:20	—	Reserved for set extension
21:26	CSET	Cache Set Specifies the cache set to be selected
27:29	—	Reserved for set/command extension
30:31	CCMD	Cache Command 00 = The data contained in this entry is invalidated 01 = Reserved 1x = Reserved

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error.

15.6.8.2 L1FINV1

The SPR number for L1FINV1 is 959 in decimal. The L1FINV1 register is shown in *Figure 94*.

0–6	7	8–19	20–26	27–29	30–31
0	CWAY	0	CSET	0	CCMD

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 959; Read/Write; Reset - 0x0

Figure 94. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in *Table 162*.

Table 162. L1FINV1 Field Descriptions

Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8:19	—	Reserved for set extension
20:26	CSET	Cache Set Specifies the instruction cache set to be selected
27:29	—	Reserved for set/command extension
30:31	CCMD	Cache Command 00 = The data contained in this entry is invalidated 01 = Reserved 1x = Reserved

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error.

15.6.9 Cache EDC/ECC Protection

Cache protection is supported for both the tag and data arrays of each cache. Tag arrays of both instruction and data caches support error detection and correction. Each doubleword in the data arrays of the ICache and for each word in the data arrays of the DCache, have single and double bit error detection.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error, to protect against internal addressing failures.

Tag ECC and data EDC checking is controlled by the $L1CSR0_{DCECE}$ and $L1CSR1_{ICECE}$ control fields. When error checking is enabled, checking is performed on each cache access.

For cache lookups due to instruction fetching, normal loads, or normal stores, if an uncorrectable tag ECC error is detected on any portion of the accessed tags, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, an error is also signaled.

Store hits to the Dcache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the Dcache during otherwise idle cycles once two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur, and one or more syndrome bits to be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled, or both may occur, depending on the error action control setting in the appropriate cache control register.

15.6.9.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the Icache and Dcache has a lockout indicator composed of a redundant set of lockout bits which can be selectively set when certain errors occur in either the tag or data portion of the cache line. Use of the lockout function and control over error conditions which cause a lockout to occur are controlled by $L1CSR\{0,1\}_{[D,I]CLOC}$. When the lockout function is enabled, lines which encounter selected tag ECC or data EDC errors on normal cacheable instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Lockout indicators may be cleared by software performing a cache invalidation operation (setting $L1CSR\{0,1\}_{[D,I]CINV}$) in conjunction with a cache lockout indicator invalidation (setting

L1CSR{0,1}$_{[D,I]CLOINV}$). Three stored redundant lockout bits are provided to allow for single and double-bit parity errors to be detected on the lockout indicators themselves.

When operating in machine check mode, if lockout controls are enabled via L1CSR{0,1}$_{[D,I]CLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception. If correction/auto-invalidation is instead enabled, on each cache lookup operation, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by re-writing all lockout bits to the asserted state, and no machine check is generated.

Lockout bit parity errors are ignored for non-cacheable accesses in either error action mode. In addition, to avoid certain exception conditions and for consistency with error reporting, non-cacheable accesses and specialized load/store accesses do not set LO bits; instead, cache contents are ignored for those errors since the cache is bypassed. A future cacheable normal access will perform the lockout function if required.

15.6.10 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting EDC/ECC errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If L1CSR1$_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant EDC/ECC check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If L1CSR0$_{DCEI}$ is set, any cache line fill to the data cache data array has the associated two most significant EDC/ECC check bits inverted in the data array for each word loaded. Additionally, inverted bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the EDC/ECC bit errors, a cache invalidation or an invalidation of the lines which have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction, or an L1FINV[0,1] invalidation operation.

15.7 Exceptions

As specified by the PowerISA 2.06 architecture, interrupts can be either precise or imprecise, synchronous or asynchronous, and critical or non-critical. Asynchronous exceptions are caused by events external to the processor’s instruction execution; synchronous exceptions are directly caused by instructions or an event somehow synchronous to the program flow, such as a context switch. A precise interrupt architecturally guarantees that no instruction beyond the instruction causing the exception has (visibly) executed. Critical interrupts are provided with a separate save/restore register pair (CSRR0/CSRR1) to allow certain critical exceptions to be handled within a non-critical interrupt handler. Machine check interrupts are also provided with a separate save/restore register pair (MCSRR0/MCSRR1) to allow machine check exceptions to be handled within a non-critical or critical interrupt handler. Debug interrupts are also provided with a separate save/restore register pair (DSRR0/DSRR1) to allow debug exceptions to be handled within a critical, non-critical, or machine check interrupt handler.

Interrupts implemented in Zen and the exception conditions that cause them are listed in *Table 163*.

Table 163. Exceptions and Conditions

Interrupt Type	Interrupt Vector Offset Value	Causing Conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	1) Reset
Critical Input	0x00(1)	**p_critint_b** is asserted and MSR_{CE}=1.
Machine check	0x10	1) **p_mcp_b** transitions from negated to asserted 2) ISI or Bus Error on first instruction fetch for an exception handler 3) EDC/ECC Error signaled on cache or Local Memory access 4) External bus error 5) Stack limit check failure
Machine check (NMI)	0x10	1) Non-Maskable Interrupt.
Data Storage	0x20	1) Access control
Instruction Storage	0x30	1) Access control
External Input	0x40(2)	Interrupt Controller interrupt and MSR_{EE}=1
Alignment	0x50	1) **lmw**, **stmw** not word aligned 2) **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3) dcbz
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$=0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$=1, MPU
EFPU Data Exception	0xA0	
EFPU Round Exception	0xB0	
TBD	0xC0-0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

15.7.1 Exception Syndrome Register

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.

Bits	0–3	4	5	6	7	8	9–11	12	13	14	15	16–23	24	25	26	27–29	30	31
Field	0	PIL	PPR	PTR	FP	ST	0	0	0	BO	0	0	SPV	0	VLEMI	0	0	0

SPR - 62; Read/Write; Reset - 0x0

Figure 95. Exception Syndrome Register

The ESR bits are defined in *Table 164*.

Table 164. ESR Bit Settings

Bit(s)	Name	Description	Associated Interrupt Type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception (For Core (z420n3), PIL is used for both illegal and unimplemented instructions)	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	--	Reserved	—
13	--	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	--	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—

Table 164. ESR Bit Settings (continued)

Bit(s)	Name	Description	Associated Interrupt Type
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

15.7.2 Machine State Register

The Machine State Register defines the state of the processor. The Zen MSR is shown in *Figure 96*.

0:5	6	7:12	13	14	15	16	17	18	19	20	21	22	23	24:25	26	27	28	29	30	31
0	0	0	0	CE	0	EE	PR	0	ME	0	0	DE	0	0	IS	DS	0	PMM	RI	0

Read/ Write; Reset - 0x0

Figure 96. Machine State Register (MSR)

The MSR bits are defined in *Table 165*.

Table 165. MSR Field Descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	—	Reserved
7:12	—	Reserved
13	—	Reserved
14	CE	Critical Interrupt Enable 0 - Critical Input interrupts are disabled. 1 - Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 - External Input interrupts are disabled. 1 - External Input interrupts are enabled.

Table 165. MSR Field Descriptions (continued)

Bit(s)	Name	Description
17	PR	Problem State 0 - The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 - The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	—	Reserved
19	ME	Machine Check Enable 0 - Asynchronous Machine Check interrupts are disabled. 1 - Asynchronous Machine Check interrupts are enabled.
20	—	Reserved
21	—	Reserved
22	DE	Debug Interrupt Enable 0 - Debug interrupts are disabled. 1 - Debug interrupts are enabled.
23	—	Reserved
24	—	Reserved
25	—	Reserved
26	—	Reserved
27	—	Reserved
28	—	Reserved
29	PMM	PMM Performance monitor mark bit. System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the Performance Monitor registers PMLCa n, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt - This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken
31	—	Reserved

15.7.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome register (MCSR) to differentiate between machine check conditions. The MCSR is shown in *Figure 97*.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCP	IC_DPERR	0	DC_DPERR	EXCP_ERR	IC_TPERR	DC_TPERR	IC_LKERR	DC_LKERR	0		NMI	MAV	MEA	U	IF	LD	ST	G	0		STACK_ERR	IMEM_PERR	DMEM_RDPERR	DMEM_WRPERR	0		BUS_IRERR	BUS_DRERR	BUS_WRERR	BUS_WRDSI	0

SPR - 572; Read/Clear; Reset - 0x0

Figure 97. Machine Check Syndrome Register (MCSR)

Table 166 describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as “write ‘1’ to clear”. Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once MSR_{ME}=1.

Table 166. Machine Check Syndrome Register (MCSR)

Bit	Name	Description	Exception Type[(1)]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved		—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the Icache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the Dcache for certain situations. May also be set on locked line refill error.	Async Mchk	—
10:11	—	Reserved		—
11	NMI	NMI input pin	NMI	—

Table 166. Machine Check Syndrome Register (MCSR) (continued)

Bit	Name	Description	Exception Type(1)	Recoverable
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV=1,MEA=1 indicates that the MCAR contains an effective address and MEA=0 indicates that the MCAR contains a physical address	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an ECC error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an ECC error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an ECC error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved		—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved		—

Table 166. Machine Check Syndrome Register (MCSR) (continued)

Bit	Name	Description	Exception Type(1)	Recoverable
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved		—

1. The Exception Type indicates the exception type associated with a given syndrome bit

   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a "1" to the bit position(s) to be cleared.

   - "Status" indicates that this bit is provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a "1" to the bit position(s) to be cleared.

   - "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a "1" to the bit position.

   - "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME}=1. If MSR_{ME}=0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a "1" to the bit position(s) to be cleared.

15.7.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the Interrupt Vector Prefix register (IVPR) to form an instruction address from which execution is to begin. The format of IVPR is shown in *Figure 98*.

Vector Base (bits 0–23)	0 (bits 24–31)

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 63; Read/Write

Figure 98. Interrupt Vector Prefix Registers (IVPR)

The IVPR fields are defined in *Table 167*.

Table 167. IVPR Register Fields

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

15.7.5 Interrupt Definitions

15.7.5.1 Critical Input Interrupt (Offset 0x00)

A Critical Input exception is signalled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

Table 168 lists register settings when a Critical Input interrupt is taken.

Table 168. Critical Input Interrupt—Register Settings

Register	Setting Description		
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
CSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME— FE0 0 DE—/0(1)	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x00 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13]**) || 2'b00 (non-autovectored)		

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

15.7.5.2 Machine Check Interrupt (Offset 0x10)

The Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome register (MCSR) to record the source(s) of machine checks, and a Machine Check Address register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in *Table 169*.

Table 169. Machine Check Interrupt - Register Settings

<table>
<tr><th>Register</th><th colspan="3">Setting Description</th></tr>
<tr><td>MCSRR0</td><td colspan="3">On a best-effort basis sets this to the address of some instruction that was executing or about to be executing when the machine check condition occurred.</td></tr>
<tr><td>MCSRR1</td><td colspan="3">Set to the contents of the MSR at the time of the interrupt</td></tr>
<tr><td>MSR</td><td>SPV 0
WE 0
CE 0
EE 0
PR 0</td><td>FP 0
ME 0
FE0 0
DE 0/—(1)</td><td>FE1 0
IS 0
DS 0
PMM 0
RI 0</td></tr>
<tr><td>ESR</td><td colspan="3">Unchanged</td></tr>
<tr><td>MCSR</td><td colspan="3">Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.</td></tr>
<tr><td>Vector</td><td colspan="3">$IVPR_{0:23}$ || 0x10</td></tr>
</table>

1. Clearing of DE is optionally supported by control in HID0.

15.7.5.3 Data Storage Interrupt (Offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

Table 170 lists register settings when a DSI is taken.

Table 170. Data Storage Interrupt—Register Settings

<table>
<tr><th>Register</th><th colspan="3">Setting Description</th></tr>
<tr><td>SRR0</td><td colspan="3">Set to the effective address of the excepting load/store instruction.</td></tr>
<tr><td>SRR1</td><td colspan="3">Set to the contents of the MSR at the time of the interrupt</td></tr>
<tr><td>MSR</td><td>SPV 0
WE 0
CE—
EE 0
PR 0</td><td>FP 0
ME—
FE0 0
DE—</td><td>FE1 0
IS 0
DS 0
PMM 0
RI—</td></tr>
<tr><td>ESR</td><td>Access:</td><td colspan="2">[ST], [SPV], VLEMI. All other bits cleared.</td></tr>
<tr><td>MCSR</td><td colspan="3">Unchanged</td></tr>
</table>

Table 170. Data Storage Interrupt—Register Settings (continued)

Register	Setting Description
DEAR	For Access Control exceptions, set to the effective address of the access which caused the violation.
Vector	$IVPR_{0:23}$ || 0x20

15.7.5.4 Instruction Storage Interrupt (Offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

Table 171 lists register settings when an ISI is taken.

Table 171. Instruction Storage Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x30		

15.7.5.5 External Input Interrupt (Offset 0x40)

An External Input exception is signalled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

Table 172 lists register settings when an External Input interrupt is taken.

Table 172. External Input Interrupt—Register Settings

Register	Setting Description
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table 172. External Input Interrupt—Register Settings (continued)

Register	Setting Description		
MSR	SPV0 WE0 CE— EE0 PR0	FP0 ME— FE00 DE—	FE10 IS0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13]**) || 2‘b00 (non-autovectored)		

15.7.5.6 Alignment Interrupt (Offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** not word aligned.
- The operand of **lwarx** or **stwcx.** not word aligned.
- The operand of **lharx** or **sthcx.** not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

Table 173 lists register settings when an alignment interrupt is taken.

Table 173. Alignment Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	[ST], VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Set to the effective address of a byte of the load or store access causing the violation.		
Vector	$IVPR_{0:23}$ || 0x50		

15.7.5.7 Program Interrupt (Offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

core will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- Instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

core will invoke a Privileged Instruction program exception on attempted execution of the following instructions when MSR_{PR}=1 (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions which specify a SPRN value with $SPRN_5$=1 (even if the SPR is undefined).

core will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *PowerISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause a illegal instruction program exception.

Table 174 lists register settings when a Program interrupt is taken.

Table 174. Program Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Illegal, Unimplemented: Privileged: Trap:	PIL, VLEMI. All other bits cleared. PPR, VLEMI. All other bits cleared. PTR, VLEMI. All other bits cleared.	
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x60		

15.7.5.8 System Call Interrupt (Offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

Table 175 lists register settings when a System Call interrupt is taken.

Table 175. System Call Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the instruction *following* the system call instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x80		

15.7.5.9 Debug Interrupt (Offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM}$=1 (internal debug mode) and MSR_{DE}=1).

Table 176 lists register settings when a Debug interrupt is taken.

Table 176. Debug Interrupt—Register Settings

Register	Setting Description		
DSRR0[(1)]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP. Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
DSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE—/0[(2)] EE—/0[(2)] PR0	FP 0 ME— FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI—

Table 176. Debug Interrupt—Register Settings (continued)

<table>
<tr><th>Register</th><th colspan="2">Setting Description</th></tr>
<tr><td>DBSR[3]</td><td>Unconditional Debug Event:
Instr. Complete Debug Event:
Branch Taken Debug Event:
Interrupt Taken Debug Event:
Critical Interrupt Taken Debug Event:
Trap Instruction Debug Event:
Instruction Address Compare:
Data Address Compare:
Debug Notify Interrupt:
Return Debug Event:
Critical Return Debug Event:
External Debug Event:
Performance Monitor Debug Event:
MPU Debug Event:
and optionally, an
Imprecise Debug Event flag</td><td>UDE
ICMP
BRT
IRPT
CIRPT
TRAP
{IAC}
{DACR, DACW}
DNI
RET
CRET
{DEVT1, DEVT2}
PMI
MPU
{IDE}</td></tr>
<tr><td>ESR</td><td colspan="2">Unchanged</td></tr>
<tr><td>MCSR</td><td colspan="2">Unchanged</td></tr>
<tr><td>DEAR</td><td colspan="2">Unchanged</td></tr>
<tr><td>Vector</td><td colspan="2">IVPR$_{0:23}$ || 0x90</td></tr>
</table>

1. assumes that the Debug interrupt is precise
2. conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

15.7.5.10 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signalled to the processor through the assertion of system-defined signals.

A System reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

Table 177 lists register settings when a System Reset interrupt is taken.

Table 177. System Reset Interrupt—Register Settings

Register	Setting Description
CSRR0	Undefined.
CSRR1	Undefined.

Table 177. System Reset Interrupt—Register Settings (continued)

Register	Setting Description		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI 0
ESR	Cleared		
DEAR	Undefined		
Vector	[**p_rstbase[0:29]**] || 2’b00		

15.7.5.11 Embedded Floating-point Data Interrupt (Offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and a EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

Table 178 lists register settings when a EFPU Floating-point Data interrupt is taken.

Table 178. Embedded Floating-point Data Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting EFPU instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	SPV, VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0xA0		

15.7.5.12 Embedded Floating-point Round Interrupt (Offset 0xB0)

The Embedded Floating-point Round interrupt is taken when a EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

Table 179 lists register settings when a EFPU Floating-point Round interrupt is taken.

Table 179. Embedded Floating-point Round Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	SPV, VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0xB0		

15.7.5.13 Performance Monitor Interrupt (Offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- $PMLCax_{CE}$ = 1; that is, for the given counter the overflow condition is enabled.
- $PMCx_{OV}$ = 1; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

Table 180 lists register settings when an performance monitor interrupt is taken.

Table 180. Performance Monitor Interrupt—Register Settings

Register	Setting Description		
SRR0/ DSRR0(1)	Set to the effective address of the next instruction to be executed.		
SRR1/ DSRR1(1)	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE—/0(2) EE 0/—(3) PR 0	FP 0 ME— FE0 0 DE—/0(4)	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		

Table 180. Performance Monitor Interrupt—Register Settings (continued)

Register	Setting Description
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x70

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1'
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ =0
4. DE is cleared if $PMGC0_{UDI}$ = 1'

15.8 MPU

15.8.1 MPU Overview

The Core (z420n3) Memory Protection Unit (MPU) provides the capability of protecting regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints

Software managed by **mpure** and **mpuwe** instructions

15.8.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MPU Assist registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with a **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with a **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

15.8.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

Figure 99. MPU Read Entry Instruction (mpure)

mpure **mpure**

mpu read entry

0–5	6	7–20	21–30	31
31	1	///	1 1 1 0 1 1 0 0 1 0	/

```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

15.8.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

Figure 100. MPU Write Entry Instruction (mpuwe)

mpuwe **mpuwe**

mpu write entry

0–5	6	7–20	21–30	31
31	1	///	1 1 1 1 0 1 0 0 1 0	/

```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

15.8.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the Core (z420n3).

Figure 101. MPU Synchronize Instruction (mpusync)

mpusync **mpusync**

MPU Synchronize

mpusync

0–5	6	7–10 / 11–15 / 16–20	21–30	31
31	1	///	1 0 0 0 1 1 0 1 1 0	/

15.8.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the Core (z420n3) MPU design. The MMUCFG register is shown in *Figure 102*.

0	RASIZE	0	PIDSIZE	NMPUS	NTLBS	MAVN
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14	15 16 17 18 19 20	21 22 23 24 25	26 27	28 29	30 31

SPR - 1015; Read-Only

Figure 102. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in *Table 181*.

Table 181. MMUCFG Field Descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000- This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 - PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 - This version of the MMU implements one MPU structure: a fully-associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 - This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 - This version of the MMU implements Version 3.1 of the MMU Architecture

15.8.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the Core (z420n3) MPU. The MPU0CFG register is shown in *Figure 103*.

0	1–7	8–11	12–15	16	17	18	19–22	23–25	26–28	29–31
FASSOC	0	MINSIZE	MAXSIZE	IPROT	0	UAMSKA	0	SHENTRY	DENTRY	IENTRY

SPR - 692; Read-Only

Figure 103. MPU0 Configuration Register (MPU0CFG)

The MPU0CFG bits are described in *Table 182*.

Table 182. MPU0CFG Field Descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 - Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size 0x0 - Smallest region size is 1 byte Note: due to the use of access address matching, the effective smallest region size is 8 bytes
12:15	MAXSIZE	Maximum Region Size 0xB - Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 - Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 - Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 - Indicates that MPU0 contains 6 shared entries
26:28	DENTRY	Number of Data Entries 0x4 - Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 - Indicates that MPU0 contains 6 dedicated Instruction entries

15.8.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in *Figure 104*.

0–15	16	17	18	19	20	21	22–23	24	25	26	27	28	29	30	31
0	BYPSR	BYPSW	BYPSX	BYPUR	BYPUW	BYPUX	0	DRDEN	DWDEN	IDEN	0	TIDCTL	0	MPUFI	MPUEN

SPR - 1014; Read/ Write; Reset - 0x0

Figure 104. MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0CSR0 bits are described in *Table 183*.

Table 183. MPU0CSR0 - MPU0 Control and Status Register 0

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses 0 - No Bypass of MPU protections for Supervisor Read accesses 1 - Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
17	BYPSW	Bypass MPU protections for Supervisor Write accesses 0 - No Bypass of MPU protections for Supervisor Write accesses 1 - Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses 0 - No Bypass of MPU protections for Supervisor Instruction accesses 1 - Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI).
19	BYPUR	Bypass MPU protections for User Read accesses 0 - No Bypass of MPU protections for User Read accesses 1 - Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
20	BYPUW	Bypass MPU protections for User Write accesses 0 - No Bypass of MPU protections for User Write accesses 1 - Bypass MPU protections for User Write accesses This bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).

Table 183. MPU0CSR0 - MPU0 Control and Status Register 0 (continued)

Bits	Name	Description
21	BYPUX	Bypass MPU protections for User Instruction accesses 0 - No Bypass of MPU protections for User Instruction accesses 1 - Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI).
22:23	—	Reserved
24	DRDEN	Data Read Debug Enable 0 - MPU debug events are disabled for data read accesses 1 - MPU debug events are enabled for data read accesses This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events.
25	DWDEN	Data Write Debug Enable 0 - MPU debug events are disabled for data write accesses 1 - MPU debug events are enabled for data write accesses This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events.
26	IDEN	Instruction Debug Enable 0 - MPU debug events are disabled for instruction accesses 1 - MPU debug events are enabled for instruction accesses This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. When enabled, instruction accesses are not blocked from generating MPU debug events.
27	—	Reserved
28	TIDCTL	TID usage Control 0 - TID comparisons performed normally 1 - no TID comparisons are performed for matching while in Supervisor mode When TIDCTL is set to ‘1’, the TID match is disabled (forced match) in Supervisor mode.
29	—	Reserved

Table 183. MPU0CSR0 - MPU0 Control and Status Register 0 (continued)

Bits	Name	Description
30	MPUFI	MPU flash invalidate 0 - No flash invalidate 1 - MPU1 invalidation operation When written to a ‘1’, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. Note that entries which have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable 0 - MPU is disabled 1 - MPU is enabled This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG=1 if in EDM and hardware owns MPU Debug entries.

15.8.9 MPU Assist Registers (MAS)

The Core (z420n3) uses four special purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in *Figure 105*. Fields are defined in *Table 184*.

Field	VALID	IPROT	SEL	0	RO	DEBUG	INST	SHD	0	ESEL	0	UAMSK	UW	SW	UX/UR	SX/SR	IOVR	GOVR (1)	1	1	I	0	G (1)	0
Bits	0	1	2–3	4	5	6	7	8	9–11	12–15	16	17–19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 624; Read/ Write; Reset - Unaffected

Figure 105. MPU Assist Registers 0 (MAS0)

1. This bit is not implemented in dedicated instruction entries, and is ignore in shared entries with INST=1

Table 184. MAS0 —MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 - This MPU entry is invalid 1 - This MPU entry is valid
1	IPROT	Invalidation Protect 0 - Entry is not protected from invalidation 1 - Entry is protected from invalidation

Table 184. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
2:3	SEL	Selects MPU for access: 00 - Reserved, no access 01 - Reserved, no access 10 - Select MPU 11 - Reserved, no access
4	—	Reserved
5	RO	Read-Only 0 - This MPU entry is writable 1 - This MPU entry is Read-only When set to ‘1’, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs.
6	DEBUG	Debug Control for Entry 0 - This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 - This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access.
7	INST	Instruction Entry 0 - This MPU entry is used for matching data accesses only 1 - This MPU entry is used for matching instruction accesses only When SHD=0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations. When SHD=1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry.
8	SHD	Shared Entry Select 0 - The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 - The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected. This bit is used to control selection of the Shared portion of the region descriptor table.
9:11	—	Reserved
12:15	ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved

Table 184. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
17:19	UAMSK	Upper Address Mask Control 000 - No upper address bits are masked, address matching uses all 32 address bits for accesses 001 - The most significant access address bit is forced to zero before matching 010 - The two most significant access address bits are forced to zero before matching 101 - The upper 5 access address bits are forced to zero before matching 11x - Reserved, do not use This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly.
20	UW	User Mode Write Permission 0 - No User mode Write permission 1 - User mode has Write permission Determines User mode Write (W) permission when INST=0. Ignored when INST=1.
21	SW	Supervisor Mode Write / Read Permission 0 - No Supervisor mode Write permission 1 - Supervisor mode has Write permission Determines Supervisor mode Write (W) permission when INST=0. Ignored when INST=1.
22	UX/UR	User Mode Execute / Read Permission 0 - No User mode Execute/ Read permission 1 - User mode has Execute/ Read permission Determines User mode Execute (X) permission when INST=1, or User mode Read (R) permission when INST=0.
23	SX/SR	Supervisor Mode Execute / Read Permission 0 - No Supervisor mode Execute/ Read permission 1 - Supervisor mode has Execute/ Read permission Determines Supervisor mode Execute (X) permission when INST=1, or Supervisor mode Read (R) permission when INST=0.
24	IOVR	Cache-Inhibit attribute Override 0 - No override of I attribute by entry 1 - Entry I bit overrides I attribute Determines whether this matching entry overrides the I bit settings of other matching entries and the Cache-Inhibited region configuration signals
25	GOVR/—	G attribute Override 0 - No override of G attribute by entry 1 - Entry G bit overrides G attribute Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignore in shared entries with INST=1.
26	—	Reserved
27	—	Reserved

Table 184. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
28	I	Cache Inhibited 0 - This region is considered cacheable 1 - This region is considered cache-inhibited This attribute may be overridden by the cache-inhibited region configuration input settings.
29	—	Reserved
30	G/—	Guarded 0 - Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 - All loads and stores to this region are performed without speculation (i.e. they are known to be required) This attribute may be overridden by the guarded region configuration input settings. This bit is not implemented in dedicated instruction entries, and is ignore in shared entries with INST=1.
31	—	Reserved

The MAS1 register is shown in *Figure 106*. Fields are defined in *Table 185*.

0	TID	0	TIDMSK
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31

SPR - 625; Read/ Write; Reset - Unaffected

Figure 106. MPU Assist Register 1 (MAS1)

Table 185. MAS1 —Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.

Table 185. MAS1 —Descriptor Context and Configuration Control (continued)

Bit	Name	Comments, or Function when Set
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx - TID[0] comparison to PID0[0] not masked 1xxxxxxx - TID[0] comparison to PID0[0] is masked x0xxxxxx - TID[1] comparison to PID0[1] not masked x1xxxxxx - TID[1] comparison to PID0[1] is masked xx0xxxxx - TID[2] comparison to PID0[2] not masked xx1xxxxx - TID[2] comparison to PID0[2] is masked xxx0xxxx - TID[3] comparison to PID0[3] not masked xxx1xxxx - TID[3] comparison to PID0[3] is masked xxxx0xxx - TID[4] comparison to PID0[4] not masked xxxx1xxx - TID[4] comparison to PID0[4] is masked xxxxx0xx - TID[5] comparison to PID0[5] not masked xxxxx1xx - TID[5] comparison to PID0[5] is masked xxxxxx0x - TID[6] comparison to PID0[6] not masked xxxxxx1x - TID[6] comparison to PID0[6] is masked xxxxxxx0 - TID[7] comparison to PID0[7] not masked xxxxxxx1 - TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID[0] comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in *Figure 107*. Fields are defined in *Table 186*.

UPPER BOUND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 626; Read/ Write; Reset - Unaffected

Figure 107. MPU Assist Register 2 (MAS2)

Table 186. MAS2 - Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in *Figure 108*. Fields are defined in *Table 187*.

LOWER BOUND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 627; Read/ Write; Reset - Unaffected

Figure 108. MPU Assist Register 3 (MAS3)

Table 187. MAS3 - Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

15.8.10 MAS Registers Summary

The MAS registers are summarized in *Figure 109*.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>MAS0</td><td>VALID</td><td>IPROT</td><td colspan="2">SEL</td><td>0</td><td>RO</td><td>DEBUG</td><td>INST</td><td>SHD</td><td colspan="3">0</td><td colspan="4">ESEL</td><td>0</td><td colspan="3">UAMSK</td><td>UW</td><td>SW</td><td>UX/UR</td><td>SX/SR</td><td>IOVR</td><td>GOVR</td><td>1</td><td>1</td><td>I</td><td>0</td><td>G</td><td>0</td></tr>
<tr><td>MAS1</td><td colspan="8">0</td><td colspan="8">TID</td><td colspan="8">0</td><td colspan="8">TIDMSK</td></tr>
<tr><td>MAS2</td><td colspan="32">UPPER BOUND</td></tr>
<tr><td>MAS3</td><td colspan="32">LOWER BOUND</td></tr>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
</table>

Figure 109. MPU Assist Registers Summary

15.9 Local memories

Note: Instruction memory (IMEM) is not present in this device.

15.9.1 Local instruction and data memory overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added to the next generation processor cores. These memories provide for low latency access (zero wait-state) similar to an instruction or data cache, but do not suffer from the overhead of cache miss or cache Writethrough operations. Instead, they provide “tightly coupled” storage.

15.9.2 Local memory control and configuration

DMEM and IMEM local memory configuration is provided by a set of privileged read-only SPRs which are accessed using **mfspr** instructions. DMEM and IMEM local memory is controlled by a set of privileged device control registers (DCRs) which are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

15.9.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the Core (z425n3) local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in *Figure 110*.

DMEM BASE ADDR	0	DECUA	DECA	0	DMSIZE	0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	20	21	22	23	24 25 26 27 28 29	30 31
CURRENT DMEM BASE ADDRESS VALUE	0	1	1	0	010000 (64 KB)	00

SPR - 694; Read-only; Privileged

Figure 110. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in *Table 188*.

Table 188. DMEMCFG0 Field Descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** low order bits of this field are driven to 0's when the DMEM size is > 4 Kbyte
20	-	Reserved
21	DECUA	DECUE - DMEM Error Correction Update Available 0 - Error Correction Update is not available 1 - Error Correction Update is available for correction of a correctable single-bit error detected on a read access DECUA indicates an error scrubbing function for the DMEM is available.
22	DECA	DECUE - DMEM Error Correction Available 0 - Error Correction is not available 1 - Error Correction is available DECA indicates an error correction function for the DMEM is available.
23	-	Reserved
24:29	DMSIZE	DMSIZE - DMEM Size 010000 - The size of the DMEM is 64 Kbyte.
30:31	-	Reserved

15.9.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

DMEM BASE ADDR	0	DBYPCB	DBYPAT	DBYPDEC	DECUE	0	DBAPD	DPEIE	DICWE	DSWCE	DDAUEC	DCECE	DSECE
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	17 18	19	20	21	22	23	24	25	26	27 28	29	30	31

DCR - 496; Read/Write; Reset(1) - 19‘b0 || p_dmem_rstcfg[8,6:7,0] || 4‘b0 || p_dmem_rstcfg[1:5]; Privileged

Figure 111. DMEM Control Register 0 (DMEMCTL0)

1. reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 189. DMEMCTL0 Field Descriptions

Bits	Name	Description
0:16	DMEM BASE ADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit.
17:18	-	Reserved
19	DBYPCB	DBYPCB - DMEM Bypass Cache Bypass CPU accesses 0 - Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port 1 - Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses which target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[8]** input.
20	DBYPAT	DBYPAT - DMEM Bypass Atomic CPU accesses 0 - Atomic CPU accesses do not bypass the DMEM CPU port 1 - Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems which support reservation or decoration functionality (or both), since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[6]** input.

Table 189. DMEMCTL0 Field Descriptions (continued)

Bits	Name	Description
21	DBYPDEC	DBYPDEC - DMEM Bypass Decorated CPU accesses 0 - Decorated Load and Store CPU accesses do not bypass the DMEM CPU port 1 - Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems which support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[7]** input.
22	DECUE	DECUE - DMEM Error Correction Update Enable 0 - Error Correction Update is disabled 1 - Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be re-written with the corrected data. (Write accesses perform this correction automatically during partial-width writes.) DECUE can be used to provide an error scrubbing function for the DMEM. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[0]** input.
23	-	Reserved
24	DBAPD	DBAPD - DMEM Base Address Port Disable 0 - The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is used to define the base address of the DMEM for CPU accesses. 1 - The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses. This bit controls usage of the external Base Address port (**p_dmem_baseaddr[0:n]**) to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses.
25	DPEIE	DPEIE - DMEM Processor Error Injection Enable 0 - Error Injection is disabled 1 - A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits (p_dchk[0:1]). DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE=0.
26	DICWE	DICWE - DMEM Imprecise CPU Write Enable 0 - Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 - Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only). DICWE may allow for increased partial-width write performance when set.

Table 189. DMEMCTL0 Field Descriptions (continued)

Bits	Name	Description
27:28	DSWCE	DSWCE - DMEM Slave port Write Check/Correct Enable 00 - Slave write data is not checked or corrected for errors. 01 - Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 - Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 - Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte writes) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or double word writes are not checked or corrected for errors. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the values of the **p_dmem_rstcfg[1:2]** inputs.
29	DDAUEC	DDAUEC - DMEM Disable Address Use in Error Check 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[3]** input.
30	DCECE	DCPECE - DMEM CPU Port ECC Enable (CPU port) 0 - End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1- End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[4]** input.
31	DSECE	DSECE - DMEM Slave port Error Checking Enable (Slave port) 0 - End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 - End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[5]** input.

15.9.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally-sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses which target DMEM address range, regardless of whether the access may bypass the DMEM based on settings in DMEMCTL0 $_{DBYPCB,}$ DBYPATF, DBYPDEC.

DSWAC3	DSWAC2	DSWAC1	DSWAC0	DSRAC3	DSRAC2	DSRAC1	DSRAC0	DUWAC3	DUWAC2	DUWAC1	DUWAC0	DURAC3	DURAC2	DURAC1	DURAC0
0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31

DCR - 498; Read/Write; Reset(1) - 32'b0; Privileged

Figure 112. DMEM Control Register 1 (DMEMCTL1)

1. reset by **m_por** and **p_reset_b**.

Table 190. DMEMCTL1 Field Descriptions

Bits	Name	Description
0:1	DSWAC3	DSWAC3 - DMEM Supervisor Write Access Control for Quadrant 3 00 - Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11
2:3	DSWAC2	DSWAC2 - DMEM Supervisor Write Access Control for Quadrant 2 00 - Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10
4:5	DSWAC1	DSWAC1 - DMEM Supervisor Write Access Control for Quadrant 1 00 - Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01

Table 190. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
6:7	DSWAC0	DSWAC0 - DMEM Supervisor Write Access Control for Quadrant 0 00 - Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2‘b00
8:9	DSRAC3	DSRAC3 - DMEM Supervisor Read Access Control for Quadrant 3 00 - Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2‘b11
10:11	DSRAC2	DSRAC2 - DMEM Supervisor Read Access Control for Quadrant 2 00 - Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2‘b10
12:13	DSRAC1	DSRAC1 - DMEM Supervisor Read Access Control for Quadrant 1 00 - Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2‘b01
14:15	DSRAC0	DSRAC0 - DMEM Supervisor Read Access Control for Quadrant 0 00 - Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2‘b00

Table 190. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
16:17	DUWAC3	DUWAC3 - DMEM User Write Access Control for Quadrant 3 00 - User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2‘b11
18:19	DUWAC2	DUWAC2 - DMEM User Write Access Control for Quadrant 2 00 - User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2‘b10
20:21	DUWAC1	DUWAC1 - DMEM User Write Access Control for Quadrant 1 00 - User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2‘b01
22:23	DUWAC0	DUWAC0 - DMEM User Write Access Control for Quadrant 0 00 - User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2‘b00
24:25	DURAC3	DURAC3 - DMEM User Read Access Control for Quadrant 3 00 - User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2‘b11

Table 190. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
26:27	DURAC2	DURAC2 - DMEM User Read Access Control for Quadrant 2 00 - User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2‘b10
28:29	DURAC1	DURAC1 - DMEM User Read Access Control for Quadrant 1 00 - User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2‘b01
30:31	DURAC0	DURAC0 - DMEM User Read Access Control for Quadrant 0 00 - User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2‘b00

15.10 End-to-End ECC support

This section describes End-to-End Error detection and correction capability (e2eECC) enhancements the Core (z425n3) to address additional diagnostic capabilities for the next generation of embedded designs.

15.10.1 e2eECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) which are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

15.10.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

0																												IDAC	IECCEN	DDAC	DECCEN
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR - 510; Read/Write; Reset - 0x0 || p_e2e_rstcfg[0:3]; Privileged

Figure 113. e2eECC Control Register 0 (E2ECTL0)

Table 191. E2ECTL0 Field Descriptions

Bits	Name	Description
0:27	-	These bits are reserved
28	IDAC	IDAC - Instruction Disable Address Checking 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion are not used for checkbit/syndrome generation for Instruction AHB accesses. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 - End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 - End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.
30	DDAC	DDAC - Data Disable Address Checking 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion are not used for checkbit/syndrome generation for Data AHB accesses. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 - End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 - End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

Enabling of e2eECC operation is performed by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a one. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for non-zero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the E2ECTL0$_{[I,D]ECCEN}$ control bits.

The E2ECTL0$_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

15.10.1.2 End-to-end ECC error generation by software

Error generation provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address or checkbit error on external bus write cycles, and then may read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus.

Note that Nexus3 accesses do not cause error injection or capturing of read checkbits, regardless of the settings of the E2EECSR0 register.

Figure 114 shows the layout of E2EECSR0.

0				RCHKBIT								0				WRC	INVC	0				WCHKBIT/CHKINVT							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 114. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 192. E2EECSR0 Field Descriptions

Bits	Name	Description
0:3	-	These bits are reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	-	These bits are reserved

Table 192. E2EECSR0 Field Descriptions (continued)

Bits	Name	Description
16:17	WRC	Write Control 00 - No write checkbit substitution is performed by this control bit 01 - Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before re-writing to 01 in order to generate a subsequent checkbit substitution operation. 10 - Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11- Reserved **Note:** checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 - No error injection is performed by this control bit 01 - Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before re-writing to 01 in order to generate a subsequent error injection operation. 10 - Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 - Reserved **Note:** checkbit substitution has priority over Error injection
20:23	-	These bits are reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits /Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 - Checkbit is driven normally 1 - Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

16 System Integration Unit Lite2 (SIUL2)

16.1 Introduction

16.1.1 Overview

The System Integration Unit Lite2 (SIUL2) provides control over all the I/O ports on this device and supports 14 ports with up to 16 bits of bidirectional, general-purpose input and output signals. It supports 24 external interrupts with trigger event configuration. *Figure 115* is the block diagram of SIUL2 and its interfaces to other system components for this device.

The SIUL2 provides the GPIO ports on the device.

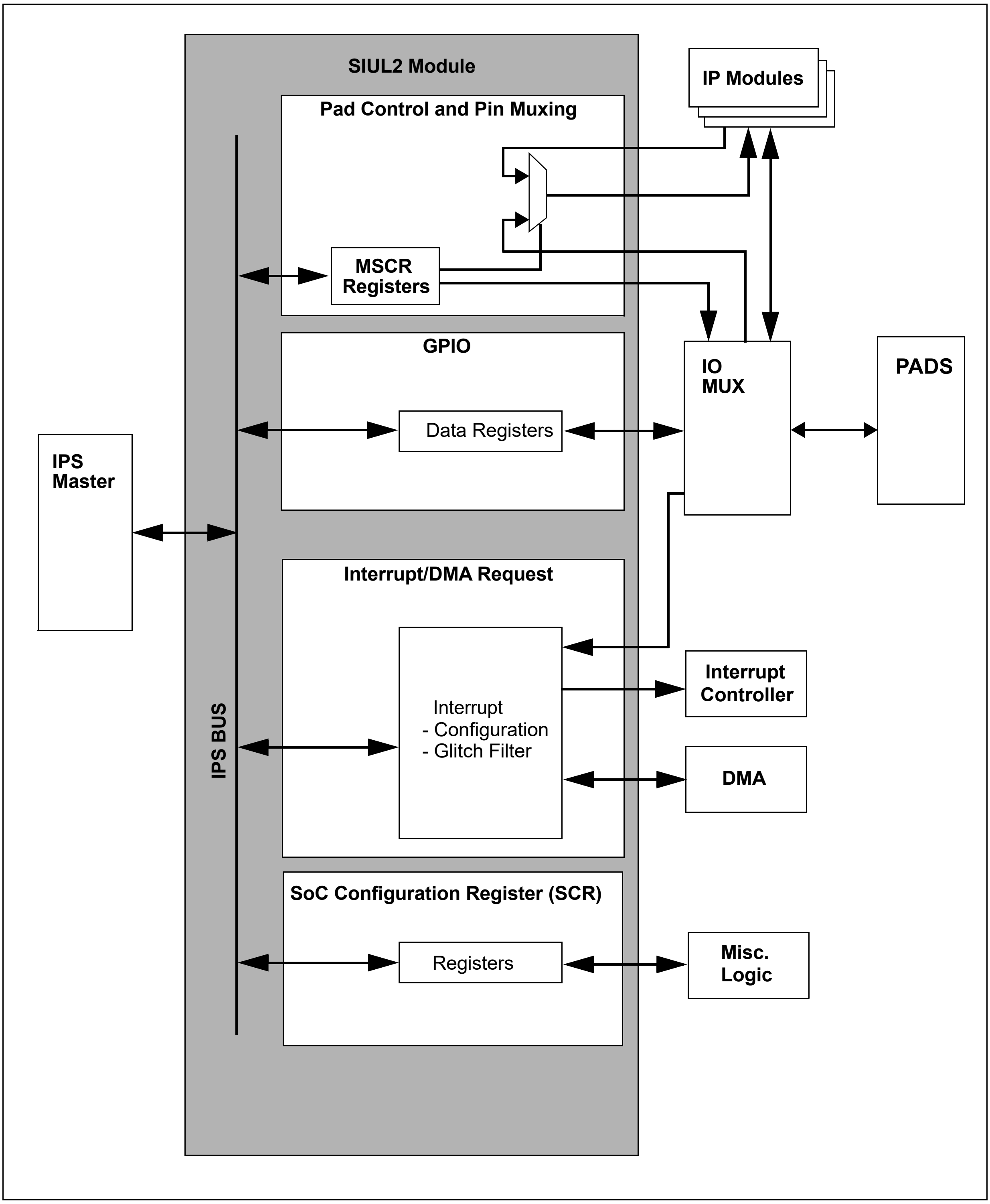
- When configured as output, you can write to the internal GPDO register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from the internal GPDI register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the driven value on the associated I/O pad.

In order to assist in software development, there are different access mechanisms to the GPIO data registers. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation:

- Access to two 16-bit ports in one access
- Read/Write access to a single bit
- A 16-bit port write with a bit mask, using single 32-bit access

There are 24 external interrupts and 32 DMA inputs to the SIUL2 that map to EIRQ pins on the device. The interrupt/DMA sources can be configured to have a digital filter to reject short glitches on the inputs.

Figure 115. SIUL2 block diagram[e]



e. For pinout details, refer to the SPC584Bx pinout Microsoft Excel file attached to the IO_Definition document.

16.1.2 Features

The SIUL2 supports these distinctive features:

- Up to 14 GPIO ports with data control[f]
  - Drive data to up to 16 independent I/O channels
  - Sample data from up to 16 independent I/O channels
- Two 16-bit registers can be read/written with one access to a 32-bit port if needed.
- External interrupt/DMA request support with:
  - 4 system interrupt vectors for 24 interrupt sources (24 connected to EIRQ pins for this device) with independent interrupt mask
  - 24 programmable digital glitch filters (one for each EIRQ)
  - Independent DMA channels for each EIRQ pin
  - Edge detection

Details about the electrical parameters and setting of functional pads can be found in the SPC584Bx pinout Microsoft Excel file attached to the IO_Definition document. These are used to configure the following pad features:

- Drive strength
- Output impedance control
- Open drain/source output enable
- Slew rate control of JTAG pins
- Internal weak pull control
- Pin function assignment
- Control of analog path switches
- Safe mode behavior configuration
- TTL, CMOS and Automotive input levels

16.1.3 Register protection

The System Integration Unit Lite2 uses the Register Protection Scheme to protect the individual registers from accidental writes. The following registers can be protected:

- SIUL2 DMA/Interrupt Request Enable Register 0(SIUL2_DIRER0)
- SIUL2 DMA/Interrupt Request Select Register 0(SIUL2_DIRSR0)
- SIUL2 Interrupt Rising-Edge Event Enable Register 0(SIUL2_IREER0)
- SIUL2 Interrupt Falling-Edge Event Enable Register 0(SIUL2_IFEER0)
- SIUL2 Interrupt Filter Enable Register 0(SIUL2_IFER0)
- SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31)
- SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)
- SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0–SIUL2_MSCR1023)

Details about the protected registers at specific addresses can be found in *Chapter 77: Register Protection (REG_PROT)* and *Chapter 7: Device configuration*.

f. For pinout details, refer to the SPC584Bx pinout Microsoft Excel file attached to the IO_Definition document.

16.2 Memory map and register description

This section provides a detailed description of all registers accessible in this module.

16.2.1 Memory map

Table 193 gives an overview of the SIUL2 registers implemented.

Note: Reserved registers will be read as 0, write will have no effect. If supported and enabled in this MCU, a transfer error will be issued when trying to access completely reserved register space. A 8-bit or 16-bit access to an "only 32-bit supported access register" gives a transfer error. Unimplemented register accesses give transfer error, that is due to register not being present.

For SIUL2_MSCR_MUX_512-SIUL2_MSCR_MUX1023, only 32/16-bit accesses are supported (8-bit writes do not occur and read zeros).

Table 193. SIUL2 memory map

Address offset	Register name	Location
0x0004	MCU ID Register #1 (SIUL2_MIDR1)	*Section 16.2.2.1*
0x0008	MCU ID Register #2 (SIUL2_MIDR2)	*Section 16.2.2.2*
0x0010	DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)	*Section 16.2.2.3*
0x0018	DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)	*Section 16.2.2.4*
0x0020	DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)	*Section 16.2.2.5*
0x0028	Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)	*Section 16.2.2.6*
0x0030	Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	*Section 16.2.2.7*
0x0038	Interrupt Filter Enable Register 0 (SIUL2_IFER0)	*Section 16.2.2.8*
0x0040–0x00BC	Interrupt Filter Maximum Counter Register 0–Interrupt Filter Maximum Counter Register 31 (SIUL2_IFMCR0–SIUL2_IFMCR31)	*Section 16.2.2.9*
0x00C0	Interrupt Filter Clock Prescaler (SIUL2_IFCPR)	*Section 16.2.2.10*
0x0100	SoC Configuration Register 0 (SIUL2_SCR0)	*Section 16.2.2.11*
0x240-0x0A3C	I/O Pin Multiplexed Signal Configuration Register 0–I/O Pin Multiplexed Signal Configuration Register 511 (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511)	*Section 16.2.2.12*
0x0A40-0x123C	I/O Pin Multiplexed Signal Configuration Register 512–I/O Pin Multiplexed Signal Configuration Register 1023 (SIUL2_MSCR0_MUX_512–SIUL2_MSCR_MUX_1023)	*Section 16.2.2.13*
0x1300–0x14FF	GPIO Pad Data Out Register 0–GPIO Pad Data Out Register 511 (SIUL2_GPDO0–SIUL2_GPDO511)	*Section 16.2.2.14*
0x1500–0x16FF	GPIO Pad Data In Register 0–GPIO Pad Data In Register 511 (SIUL2_GPDI0–SIUL2_GPDI511)	*Section 16.2.2.15*
0x1700–0x173E	Parallel GPIO Pad Data Out Register 0–Parallel GPIO Pad Data Out Register 12 (SIUL2_PGPDO0–SIUL2_PGPDO12)	*Section 16.2.2.16*

Table 193. SIUL2 memory map (continued)

Address offset	Register name	Location
0x1740–0x177E	Parallel GPIO Pad Data In Register 0–Parallel GPIO Pad Data In Register 12 (SIUL2_PGPDI0–SIUL2_PGPDI12)	*Section 16.2.2.17*
0x1780–0x17FC	Masked Parallel GPIO Pad Data Out Register 0–Masked Parallel GPIO Pad Data Out Register 12 (SIUL2_MPGPDO0–SIUL2_MPGPDO12)	*Section 16.2.2.18*

16.2.2 Register descriptions

This section describes in address order all the SIUL2 registers.

16.2.2.1 MCU ID Register #1 (SIUL2_MIDR1)

This register contains identification information about the device.

In case of HPREG configuration, the reset values are:

- QFP176: 0x584244xx
- QFP144: 0x584234xx
- QFP100: 0x584224xx
- QFP64: 0x584204xx

In case of DREG configuration, the reset values are:

- QFP176: 0x584246xx
- QFP144: 0x584236xx
- QFP100: 0x584226xx
- QFP64: 0x584206xx

Note: The “x” value in the reset values above listed depends on the minor device silicon version.

Offset: 0x0004 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	0	1	0	1	1	0	0	0	0	1	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PKG					BALL_CONF	0	MAJOR_MASK				MINOR_MASK			
W																
Reset	0	0[1]	1[1]	0[1]	0[1]	1[1]	1	0	X[2]	X[2]	X[2]	X[2]	X[2]	X[2]	X[2]	X[2]

1. Only possible to change value by modifying the side band signals (in the Test Flash). Values are set at factory and cannot be modified.
2. Value is set at factory and will vary each revision of device

Figure 116. SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

Table 194. SIUL2_MIDR1 field descriptions

Field	Description
0:15 PARTNUM	MCU Part Number These digits identify the part number of the chip. Default is 0x5842
17:21 PKG	Package Settings Can by read by software to determine the package type that is used for the particular device: 00001 QFP64 01001 QFP100 01101 QFP144 10001 QFP176 All other combinations are reserved for future use.
22 BALL_CONF	0 device configured in external ballast mode (HPREG Mode) 1 device configured in internal ballast mode (DREG Mode)
24:27 MAJOR_MASK	Major Mask revision
28:31 MINOR_MASK	Minor Mask revision

16.2.2.2 MCU ID Register #2 (SIUL2_MIDR2)

This register contains identification information about the device.

Offset: 0x0008 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE_1				FLASH_SIZE_2				0	0	0	0	0	0	0
W																
Reset	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FAMILYNUM								0	0	0	0	0	0	0	0
W																
Reset	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 117. SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

Table 195. SIUL2_MIDR2 field description

Field	Description
0 SF	Manufacturer: 0 Reserved 1 STMicroelectronics
1:4 FLASH_SIZE_1	Coarse granularity for Flash memory size Needs to be combined with FLASH_SIZE_2 to calculate the actual memory size: 0000 16 KB 0001 32 KB 0010 64 KB 0011 128 KB 0100 256 KB 0101 512 KB 0110 1 MB 0111 2 MB 1000 4 MB 1001 8 MB 1010 16 MB 1011 32 MB 1100 64 MB 1101 128 MB 1110 256 MB 1111 512 MB
5:8 FLASH_SIZE_2	Fine granularity for Flash memory size Needs to be combined with FLASH_SIZE_1 to calculate the actual memory size: 0000 0 KB 0001 (FLASH_SIZE_1) / 8 0010 2 * (FLASH_SIZE_1) / 8 0011 3 * (FLASH_SIZE_1) / 8 0100 4 * (FLASH_SIZE_1) / 8 0101 5 * (FLASH_SIZE_1) / 8 0110 6 * (FLASH_SIZE_1) / 8 0111 7 * (FLASH_SIZE_1) / 8 1000 8 * (FLASH_SIZE_1) / 8 1001 9 * (FLASH_SIZE_1) / 8 1010 10 * (FLASH_SIZE_1) / 8 1011 11 * (FLASH_SIZE_1) / 8 1100 12 * (FLASH_SIZE_1) / 8 1101 13 * (FLASH_SIZE_1) / 8 1110 14 * (FLASH_SIZE_1) / 8 1111 15 * (FLASH_SIZE_1) / 8
16:23 FAMILYNUM	ASCII character in MCU part number: 0x42 “B” All other values reserved for future use.

16.2.2.3 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)

The DMA/Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set irrespective of the corresponding DMA/Interrupt Request Enable bit in DMA/Interrupt Request Enable Register (SIUL2_DIRER0) is enabled. The EIF bit remains set until cleared by software or through the servicing of a DMA request. The EIF bits are cleared by writing a ‘1’ to the bits. A write of ‘0’ has no effect.

This register contains the interrupt flags.

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	EIF23	EIF22	EIF21	EIF20	EIF19	EIF18	EIF17	EIF16
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 118. SIUL2 DMA/Interrupt Status Flag Register (SIUL2_DISR0)

Table 196. SIUL2_DISR0 field descriptions

Field	Description
8:31 EIF[23:0]	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

16.2.2.4 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

The DMA/Interrupt Request Enable Register enables the assertion of DMA or interrupt request if the corresponding External IRQ Flag bit is set in SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0). This register enables the interrupt messaging to the interrupt controller.

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	EIRE23	EIRE22	EIRE21	EIRE20	EIRE19	EIRE18	EIRE17	EIRE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 119. SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

Table 197. SIUL2_DIRER0 field descriptions

Field	Description
8:31 EIRE[23:0]	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR

16.2.2.5 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

The DIRSR selects between the DMA or interrupt request. If the corresponding bits are set in SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0) and SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0), then the DMA/Interrupt Request Select bit determines whether DMA or an interrupt request is asserted. EIRQ are the external interrupt package pins on the device.

Offset: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DIRS23	DIRS22	DIRS21	DIRS20	DIRS19	DIRS18	DIRS17	DIRS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	DIRS15	DIRS14	DIRS13	DIRS12	DIRS11	DIRS10	DIRS9	DIRS8	DIRS7	DIRS6	DIRS5	DIRS4	DIRS3	DIRS2	DIRS1	DIRS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 120. SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

Table 198. SIUL2_DIRSR0 field descriptions

Field	Description
8:31 DIRS[23:0]	DMA/Interrupt Request Select Register Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected

16.2.2.6 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	IREE23	IREE22	IREE21	IREE20	IREE19	IREE18	IREE17	IREE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IREE15	IREE14	IREE13	IREE12	IREE11	IREE10	IREE9	IREE8	IREE7	IREE6	IREE5	IREE4	IREE3	IREE2	IREE1	IREE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 121. SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

Table 199. SIUL2_IREER0 field descriptions

Field	Description
8:31 IREE[23:0]	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

16.2.2.7 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

Note: If both IREE bit and IFEE bit are cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	IFEE23	IFEE22	IFEE21	IFEE20	IFEE19	IFEE18	IFEE17	IFEE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	IFEE15	IFEE14	IFEE13	IFEE12	IFEE11	IFEE10	IFEE9	IFEE8	IFEE7	IFEE6	IFEE5	IFEE4	IFEE3	IFEE2	IFEE1	IFEE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 122. SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

Table 200. SIUL2_IFEER0 field descriptions

Field	Description
8:31 IFEE[23:0]	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

16.2.2.8 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs.

Offset: 0x0038 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	IFE23	IFE22	IFE21	IFE20	IFE19	IFE18	IFE17	IFE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	IFE15	IFE14	IFE13	IFE12	IFE11	IFE10	IFE9	IFE8	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 123. SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

Table 201. SIUL2_IFER0 field descriptions

Field	Description
8:31 IFE[23:0]	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

16.2.2.9 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

These registers are used to configure the filter counter associated with each digital glitch filter.

Offset: 0x0040 + n * 0x4 (n = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MAXCNT			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 124. SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

Table 202. SIUL2_IFMCR*n* field descriptions

Field	Description
28:31 MAXCNT	Maximum Interrupt Filter Counter setting (MAXCNTx) Filter Period = TCK*MAXCNTx + n*TCK where (n = -1 to 3) MAXCNT can be 0 to 15 (for MAXCNT<3 filter will behave as all PASS filter) TCK is the Prescaled filter clock period, which is the IRC clock prescaled to IFCP value, specified in SIUL2_IFCPR T(IRC): Basic filter clock period 62.5 ns (f = 16 MHz) Note that filter delay is 2 system clock cycles more than filter period. In general, Filter Period and Filter delay are not integer multiple of TCK because input to Filter is asynchronous whereas output is synchronous with respect to the TCK. T(DELAY) = (1 to 5) * TCK + MAXCOUNT * TCK. The SW is to expect the worst filter delay value

16.2.2.10 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

This register configures a clock prescaler that selects the clock for all digital filters. The prescaler is applied to the input clock to the SIUL2, which is the system PBRIDGE clock counter in the SIUL2. Refer to *Chapter 25: Clocking* for information on clocking.

Offset: 0x00C0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IFCP			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 125. SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

Table 203. SIUL2_IFCPR field descriptions

Field	Description
28:31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15.

16.2.2.11 Soc Configuration Register 0 (SIUL2_SCR0)

The ETHERNET_0_MODE bit is used to select between the Media Independent Interface (MII) and the Reduced Media Independent Interface (RMII) mode for the Ethernet.

MASK_OPCn is used to prevent the SSWU OPC channel n to drive the pad during standby.

PADn_MUX_SEL is used to maintain the pad value at standby exit (PADn_MUX_SEL = 0)

Note: PADn_MUX_SEL must be set to 1 to be able to change the pads value and direction after stand-by while system is in the RUN mode.

Offset: 0x0100 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETHERNET_0_MODE	0	0	0	0	0	0	0	Reserved(1) 0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PAD53_OPC_MASK	PAD92_OPC_MASK	PAD91_OPC_MASK	PAD90_OPC_MASK	PAD89_OPC_MASK	PAD24_OPC_MASK	PAD25_OPC_MASK	0	PAD53_MUX_SEL	PAD92_MUX_SEL	PAD91_MUX_SEL	PAD90_MUX_SEL	PAD89_MUX_SEL	PAD24_MUX_SEL	PAD25_MUX_SEL	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 126. Soc Configuration Register 0 (SIUL2_SCR0)

1. This bit shall be left at '0' default value by application.

Table 204. SIUL2_SCR0 field descriptions

Field	Description
0 ETHERNET_0_MODE	Ethernet Controller mode setting 0 RMII is the default state. 1 MII is selected. **Note:** MII/RMII pad configuration is done independently in the SIUL2 MSCR registers.
16 PAD53_OPC_MASK	PAD[53] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
17 PAD92_OPC_MASK	PAD[92] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
18 PAD91_OPC_MASK	PAD[91] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
19 PAD90_OPC_MASK	PAD[90] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
20 PAD89_OPC_MASK	PAD[89] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
21 PAD24_OPC_MASK	PAD[24] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
22 PAD25_OPC_MASK	PAD[25] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
24 PAD53_MUX_SEL	PAD[53] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
25 PAD92_MUX_SEL	PAD[92] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
26 PAD91_MUX_SEL	PAD[91] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
27 PAD90_MUX_SEL	PAD[90] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value

Table 204. SIUL2_SCR0 field descriptions (continued)

Field	Description
28 PAD89_MUX_SEL	PAD[89] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
29 PAD24_MUX_SEL	PAD[24] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
30 PAD25_MUX_SEL	PAD[25] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value

16.2.2.12 I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO*n*)

Refer to *Section 16.2.3: Multiplexed Signal Configuration Registers* for an introduction to the MSCRs.

This section defines the I/O pin MSCRs. These registers select the output function on the associated I/O pin. The I/O MSCRs control the drive strength, output drive circuit, pull up/down, input buffer enable, input hysteresis, input level selection, and safe-mode operation of the associated I/O pin.

Note: Not all pads have all bits, such as only some pads have hysteresis. Refer to the I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document.

Note that when an I/O pin is configured as a single-ended input, the input buffer must be enabled for the pin using the MSCR[IBE] bit. This enables the input to all input destinations connected to the pin, including the GPIO input. In this way, the GPIO can be read at any time, regardless of the pin function. For IP blocks that have multiple input sources, the input source is selected in the multiplexed input selection MSCRs, and the associated I/O pin MSCR for the selected input must have the MSCR[IBE] bit enabled.

All bits and fields in the I/O MSCRs are applicable for all GPIO ports on the device. This includes all ports on the device with digital output function. For ADC input ports with digital input only (no output), the OERC, ODC, SMC, INV and SSS bits/fields do not apply. The SSS is always 8b0 for input only ports, enabling the general purpose input. The analog input path is enabled in the ADC logic when the channel is selected for conversion.

The MSCR reset state is detailed in the I/O Signal Table of the SPC584Bx pinout Microsoft Excel file attached to the IO_Definition document.

Offset: 0x0240 + n * 0x4 (n = 0 to 511) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	OERC		0	0	ODC		SMC	0	ILS		IBE	0	WPDE	WPUE
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0	0	0	0	0	0	0	0	0	0	0	SSS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 127. I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO*n*)

Table 205. SIUL2_MSCR_IO*n* field description

Field	Description
2:3 OERC	Output Edge Rate Control Specifies the output impedance, drive strength and slew rate of the associated pin. Refer to output Impedance columns in the I/O Signal Description Table for the applicable edge rates for each I/O pin on the device. The I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) are attached to the IO_Definition document. Refer to the device Data Sheet for the electrical characteristics of the weak, medium, strong, and very strong pad types **Note:** OERC[2] programming is required for forward compatibility. 00 Weak drive 01 Medium drive 10 Strong drive 11 Very Strong drive
6:7 ODC	Output Drive Control Specifies the type of output drive control for the associated pin. 00 Output buffer disabled 01 Open-drain 10 Push-pull 11 Open-source
8 SMC	Safe Mode Control Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode. Resets to 0 except for PB[11] ERROR0 pin which resets to 1. 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Does not disable

Table 205. SIUL2_MSCR_IO*n* field description (continued)

Field	Description
10:11 ILS	Input Level Selection Specifies the logic family for the associated pin, which determines its logic switching levels. Refer to the device Data Sheet for the electrical characteristics of the I/O pad input buffer types. Refer as well to the IO_Definition Microsoft Excel file. 00 Reserved 01 TTL (CMOS for LP pads) 10 LVDS 11 CMOS
12 IBE	Input Buffer Enable Enables the associated pin's input buffer. 0 Disabled 1 Enabled
14 WPDE	Weak Pulldown Enable Used only when the associated destination is a chip pin. Enables the associated pin's weak pulldown resistor. If pin has analog input-only functionality: when WPDE and WPUE are both enabled that pin is configured with Strong pulldown. On all other pins (including analog bidirectional) this configuration means neither a pullup nor a pulldown. 0 Disabled 1 Enabled
15 WPUE	Weak Pullup Enable Used only when the associated destination is a chip pin. Enables the associated pin's weak pullup resistor. If pin has analog input-only functionality: when WPDE and WPUE are both enabled that pin is configured with Strong pulldown. On all other pins (including analog bidirectional) this configuration means neither a pullup nor a pulldown. 0 Disabled 1 Enabled
16 INV	Invert The output selected by the corresponding MSCR SSS field can be inverted before it is driven on the I/O pin with the INV bit. **Note:** Use of the INV bit is not supported when the ODC is configured for open-drain or open-source modes, and the output data will not be as expected for an inverted open-drain or open-source connection. 0 The output selected by the SSS field is not inverted before being driven to the pin 1 The output selected by the SSS field is inverted before being driven to the pin
28:31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. Refer to the *Section 16.2.3.1* and *Section 16.2.3.2*.

Table 206. I/O MSCR Reset State Exceptions

Port	Critical Function	SMC	WPDE	WPUE
PA[4]	—	0	0	
PA[5]	JCOMP	1		
PA[6]	TCK	1	1	0
PA[7]	TMS	1	0	1
PA[8]	TDI	0	0	1
PA[9]	TDO	0	0	1(1)
PA[14]	—	0	0	
PB[11]	ERROR	1	0	0
PC[10:15]	—	0	0	
PD[6:7]	—	0	0	
PE[12]	—	0	0	
PF[13]	—	0	0	
PI[14:15]	—	0	0	
PK[12:15]	—	0	0	
PM[0:8]	—	0	0	

1. When JTAG is enabled (JCOMP asserted) during reset, the TDO pin is pulled up during reset, but Hi-Z after. If JTAG is disabled, the TDO pin defaults to input/pull-up during and after reset.

16.2.2.13 Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX*n*)

Refer to *Multiplexed Signal Configuration Registers* for an introduction to the MSCRs.

This section defines the multiplexed input selection MSCRs. These registers select the input source for IP blocks on the device that have inputs from multiple sources, both other IP blocks and I/O pins. It is possible to configure the pad to be in input without having to connect it externally. This is possible, either by enabling internal pull-up/pull-down or disabling the input buffer (but in this case data is seen as 0 internally).

Offset: 0x240 + *n* * 0x4 (*n* = 512 to 1023) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0	0	0	0	0	0	0	0	0	0	0	SSS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 128. Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX*n*)

Table 207. SIUL2_MSCR_MUX*n* field descriptions

Field	Description
16 INV	Invert The input source signal selected by the corresponding SSS field for this MSCR can be inverted before it connects to the destination input with the INV bit. 0 SSS selected input signal is connected directly to the destination input 1 SSS selected input signal is inverted before it is connected to the destination input
28:31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. Refer to *Section 16.2.3.1* and *Section 16.2.3.2*.

16.2.2.14 GPIO Pad Data Output Registers (SIUL2_GPDO*n*)

These registers can be used to set or clear a single GPIO pad with a byte access. For mapping of ports to GPDO registers, refer to the I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document. Note that the GPDO registers do not apply for ports PA[5], PA[6] and PA[7]. The PA ports are JTAG pins with no GPIO. Writes to the GPDO register corresponding to these ports have no effect.

Note: For the exact number of registers, refer to the SPC584Bx pinout Microsoft Excel file attached to the IO_Definition document.

Offset: 0x1300 + n * 0x1 (n = 0 to 511) Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDO
W								
Reset	0	0	0	0	0	0	0	0

Figure 129. SIUL2 GPIO Pad Data Output register (SIUL2_GPDO*n*)

Table 208. SIUL2_GPDO*n* field descriptions

Field	Description
7 PDO	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

16.2.2.15 GPIO Pad Data Input Registers (SIUL2_GPDI*n*)

These registers can be used to read the GPIO pad data with a byte access. For mapping of ports to GPDI registers, refer to the I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document. Note that the GPDI registers do not apply for ports PA[5], PA[6] and PA[7]. These three PA ports are JTAG pins with no GPIO. The GPDI values corresponding to these ports always read a value of zero.

Note: For the exact number of registers, refer to the SPC584Bx pinout Microsoft Excel file attached to the IO_Definition document.

Offset: 0x1500 + n * 0x1 (n = 0 to 511) Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDI
W								
Reset	0	0	0	0	0	0	0	0

Figure 130. SIUL2 GPIO Pad Data Input register (SIUL2_GPDI*n*)

Table 209. SIUL2_GPDI*n* field descriptions

Field	Description
7 PDI	Pad Data In This bit stores the value of the external GPIO pad associated with this register. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

16.2.2.16 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write. Note that the PGPDO registers do not apply for ports PA[5], PA[6] and PA[7]. These PA ports are JTAG pins with no GPIO. Writes to the PGPDO register corresponding to these ports have no effect.

Note: The SIUL2_PGPDO registers access the same physical resources as the SIUL2_PDO and SUIL2_MPGPDO address locations. Some examples of the mapping:

- PPDO[0][0] = PDO[0]
- PPDO[2][0] = PDO[32]
- PPDO[31][15] = PDO[511]

Offset: 0x1700 + n*0x2 (n=0 to 12) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PPDO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 131. SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO)

Table 210. SIUL2_PGPDO*n* field descriptions

Field	Description
0:15 PPDO	Parallel Pad Data Out Write or read the data register that stores the value to be driven on the pad in output mode. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO). The x and y bit indexes define which PPDO register bit is equivalent to which PDO register bit according to the following equation: PPDO[x][y] = PDO[(x*16)+y]

16.2.2.17 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read. Note that the GPDI registers do not apply for ports PA[5], PA[6] and PA[7]. These ports are JTAG pins with no GPIO. The GPDI values corresponding to these ports always read a value of zero.

Offset: 0x1740 + n*0x2 (n=0 to 12) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PPDI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 132. SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI)

Table 211. SIUL2_PGPDI*n* field descriptions

Field	Description
0:15 PPDI	Parallel Pad Data In Reads the current pad value. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and y bit indexes define which PPDI register bit is equivalent to which PDI register bit according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

16.2.2.18 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

This register can be used to selectively modify the pad values associated to PPDO[x][0:15]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read access will return 0. Note that the MPGPDO registers do not apply for ports PA[5], PA[6] and PA[7]. These are JTAG pins with no GPIO. Writes to the MPGPDO register corresponding to these ports have no effect.

Offset: 0x1780 + n*0x4 (n=0 to 12) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																0															
W	MASK																MPPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 133. SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO)

Table 212. SIUL2_MPGPDO*n* field descriptions

Field	Description
0:15 MASK	Mask Field Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0 The associated bit value in the MPPDO[x] field is ignored 1 The associated bit value in the MPPDO[x] field is written
16:31 MPPDO	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in output mode. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO). The x and y bit indexes define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][y] = PDO[(x * 16) + y]

16.2.3 Multiplexed Signal Configuration Registers

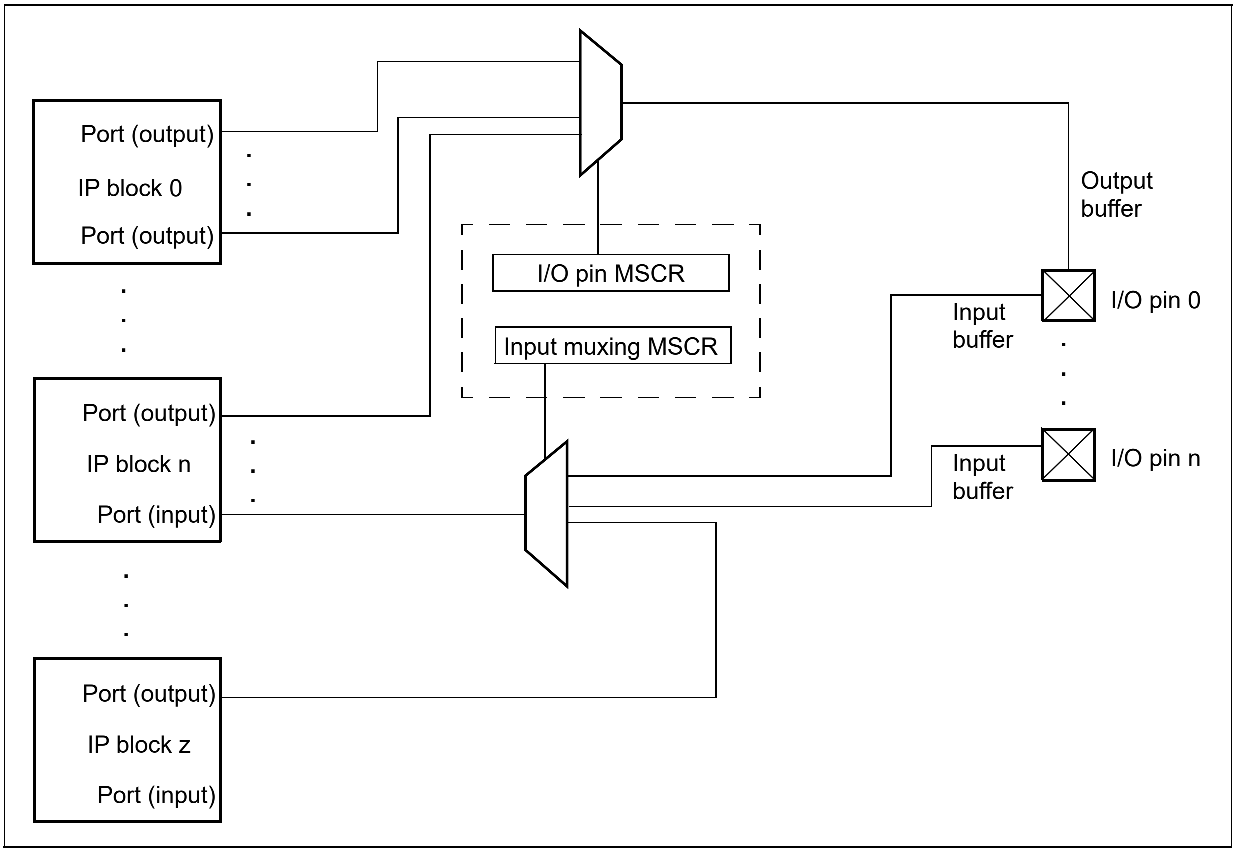
The MSCRs control the I/O pin function and electrical properties for each pin on the device, and are used to select the input signal for IP blocks on the device that have multiple input sources. There is a dedicated MSCR register for each I/O pin on the device. Each MSCR register is independent of the others, except in the case of an LVDS configuration, where two MSCR registers must be configured together in this mode. When LVDS is configured, CMOS pad is disabled (ibe and obe = 0) and when LVDS is not configured, (either of two MSCRs not configured for lvds) then the lvds pad is disabled (lvds_ibe and lvds_obe = 0).

The MSCRs are divided into two sets:

- I/O pin control registers, refer to *I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IOn)*
- Multiplexed input selection registers, refer to *Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUXn)*

The signal data flow from IP blocks on the device to and from the I/O pins, and from IP block to IP block is given in *Figure 134*.

Figure 134. MSCR I/O pin and IP block port connectivity



16.2.3.1 Chip-pin MSCR assignments and related information

For device pin MSCR register definitions, refer to the SPC584Bx I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document.

16.2.3.2 Module-port MSCR assignments and SSS values

For each internal module port that is or can be configured as an input and has more than one possible source. Refer to the SPC584Bx I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document. The Input Multiplexing Table shows the assigned MSCR, and the SSS values and their corresponding source signals.

16.3 Functional description

16.3.1 General

This section provides a complete functional description of the SIUL2.

16.3.2 Pad control

The SIUL2 controls the electrical characteristics of the pads on the device through the MSCR registers. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2.

The required controls are:

- Slew rate control
  - This is needed to offer improved EMC performance.
  - Support for a fast and slow slew rate.
- Output Impedance Control
  - 4 output impedances, drive strength and slew rates supported.
- Internal weak pull capability
  - This is needed to offer flexibility in the device configuration and the elimination of external hardware in some cases.
  - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down or no-pull enabled.
- Safe Mode behavior configuration
- Open drain/source enable
  - Needed to support different pads muxed (for example mux IIC, with non open drain pads).
- Pin Function Assignment
  - This setting defines which chip function has control over the output of the pad.

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for the user software to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register and allowing simplified duplication of software for each pad with indexed changes for each pad.

16.3.3 General purpose input or output pads (GPIO)

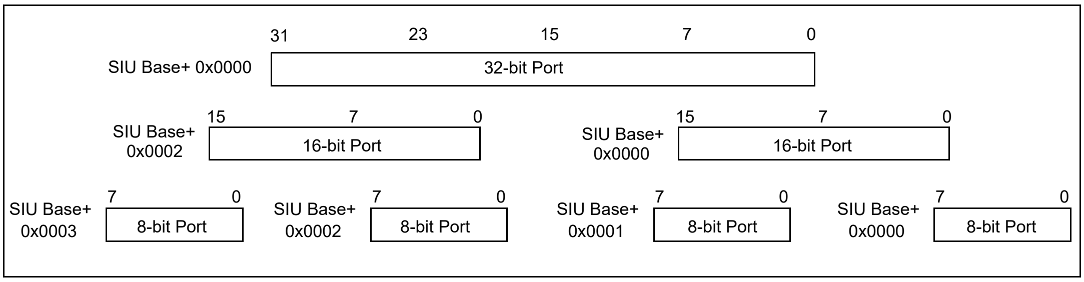
The SIUL2 allows each pad to be configured as either a General Purpose Input Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage 202 GPIO pads (205 pads less 3 reserved for JTAG) organized as ports that can be accessed for data reads and writes as 32-bit, 16-bit or 8-bit.

As shown in *Figure 135*, all port accesses are identical with each read or write being performed only at a different location to access a different port width.

Figure 135. Data Port example arrangement showing configuration for different port width accesses



This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non-GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

All general purpose pads are implemented as bidirectional.

Note: In case the bidirectional operation impacts some performances (for example ADC accuracy), or is not required for a specific pad function it is acceptable to limit the functionality of some pads to “Input Only”.

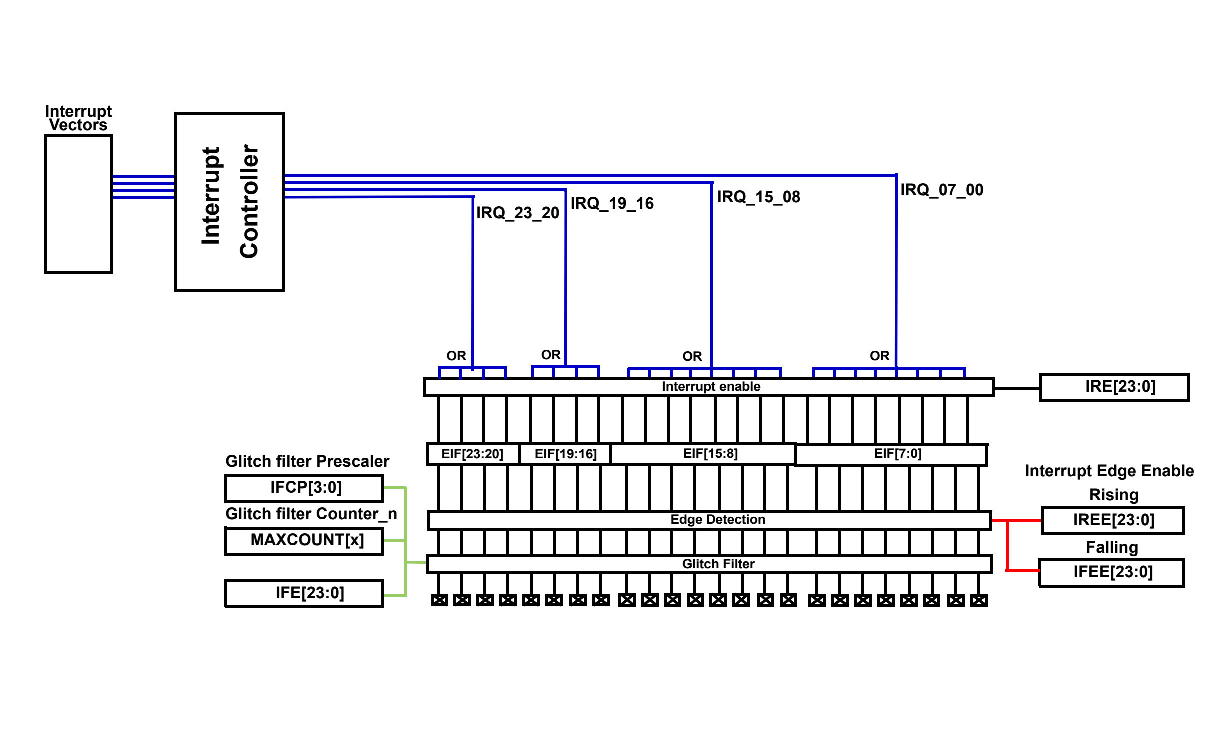
16.3.4 External interrupts/DMA requests (EIRQ pins)

The SIUL2 supports 24 external interrupts which are allocated to EIRQ pins on the device.

The SIUL2 supports 4 interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support eight or four external interrupt sources from the device pads.

Refer to *Figure 136* for an overview of the external interrupt implementation.

Figure 136. External interrupt pad diagram



All the external interrupt pads within a single group (maximum 8 pads) have equal priority. It is the responsibility of user of the software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

An MCU-specific number of external interrupt lines can have digital glitch filters applied to them. The supported range is 1 to 8. The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on an interrupt line.

16.3.4.1 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request, during pin interrupt initialization, the user must do the following:

- Mask interrupts by clearing the EIREn bits in DIRER0.
- Select the pin polarity by setting the appropriate IREEn bits in IREER0 and the appropriate IFEEn bits in IFEER0 as desired.
- Configure the appropriate bits in the MSCR[0–511] register for the external interrupt pin(s) desired as follows:
  – Clear the ODC bits to do disable output.
  – Set the IBE bit to enable the pin’s input buffer.
  – If using the internal weak pullup/pulldown, configure the appropriate WPUE and WPDE bits.

Note: MSCR_SSS bits do not need to be changed for either MSCR[0:511] or MSCR[512:1023] since the external interrupt pin input is directly connected to the SIUL for EIRQ pins.

External interrupt pins should never be configured as outputs (MSCR_ODC bits are not zeros) when external interrupt inputs are desired since false interrupts could be detected (such as from a GPIO configuration).

- Select the request desired between DMA or Interrupt by writing the appropriate DIRSn bits in DIRSR0.

Note: Not all EIRQ pins have DMA capability. Refer to Section 16.2.2.5: DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0).

- Select the desired glitch filter setup for the pins by writing the following:
  – Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMCn register for the respective external interrupt that is being used.
  – Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR register.
  – Then enable the glitch filter for the desired external interrupt pins by setting the appropriate IFEn bits in IFER0.

Note: The glitch filter clock runs on the internal RC oscillator clock whereas the SIUL logic runs on the PBRIDGEx clock. The IRC clock is at a fixed frequency of 16 MHz. The PBRIDGEx clock can on higher clock (refer to the Clocking chapter)

The glitch filter is enabled on IRC clock, so make sure at least one IRC clock occurs after enabling it before trying to detect glitchless events.

- Write to EIFn bits in DISR0 to as desired to clear any flags (for DMA it is self clearing).
- Enable the interrupt pins by setting the appropriate EIREn bits in DIREER0.

16.3.4.2 External interrupt management

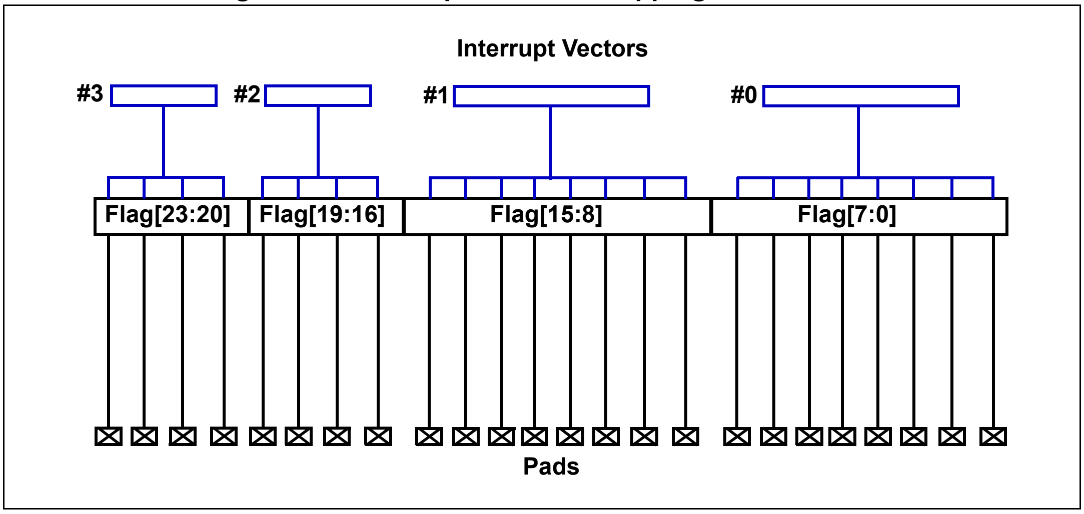
Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (SIUL2_DIRER0). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled is reserved and should not be configured.

The active IRQ edge is controlled by the users through the configuration of the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (DISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

The SPC584Bx device supports 4 SIUL2 interrupt vectors aggregating a total of 24 external interrupt sources as shown in Figure 137.

Figure 137. Interrupt to vector mapping at MCU level



16.3.4.3 External interrupt request

The EIRQ input pins on the device are sources for interrupt or DMA requests. The 24 EIRQ pins for the device map to 4 interrupt requests. The mapping of interrupt requests and EIRQ pins is given in *Table 213*.

Table 213. SIUL input trigger connections to the INTC channels

External interrupt number	SIUL IRQ vector
0 to 7	0
8 to 15	1
16 to 19	2
20 to 23	3

16.3.4.4 Interrupt Vector

The 24 EIRQ pins on the device map to an Interrupt Vector in the Interrupt Controller.

Note: Refer to INTC configuration section in the Chapter Device Configuration for the Interrupt Vector table.

16.3.4.5 DMA requests

Nine out of 16 EIRQ pins on the device map to an independent DMA request channel in the DMA controller.

Note: Refer to the DMA controller configuration section in the Chapter Device Configuration for DMA request mapping table.

17 Crossbar switch (XBAR)

17.1 Introduction

This chapter provides information on the layout, configuration, and programming of the crossbar switch. The crossbar switch connects bus masters and bus slaves using a hardware interconnect matrix. This structure allows all bus masters to access different bus slaves simultaneously with no interference while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

17.2 Features

The crossbar switch includes these distinctive features:

- Symmetric crossbar bus switch implementation
  - Concurrent accesses from different masters to different slaves
  - Configurable slave arbitration attributes on a slave-by-slave basis
- 64-bit datapath width
- Support for 8-, 16-, and 64-bit single transfers
- Support for a variety of 4-, 8-, and 16-beat burst transfers including a 4-beat, 64-bit burst for cache line accesses
- Operation at one-to-one clock frequency with the bus masters
- Support for low-power park mode

17.3 Memory map and register definition

Each slave port of the crossbar switch contains configuration registers. Read and write transfers of the configuration registers require four system bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The slave registers also feature a bit that, when set, prevents the registers from being written. The registers remain readable, but future write attempts have no effect on the registers and are terminated with a bus error response to the master initiating the write. The core, for example, takes a data storage interrupt.

Note: This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. Refer to Section 7.3.2: Crossbar switch in Chapter 7: Device configuration for the exact master/slave assignments for your device. Additionally, all references to the crossbar switch registers are based on the physical port connections, not the logical port numbers.

Table 214. XBAR memory map

Address offset (hex)	Register name	Location
SLAVE channel *n* (*n* = 0 to 7)		
0x0000 + *n**0x100	XBAR Priority Register Slave *n* (XBAR_PRS*n*)	*Section 17.3.1*
0x0010 + *n**0x100	XBAR Control Register *n* (XBAR_CRS*n*)	*Section 17.3.2*
MASTER channel *n* (*n* = 0 to 7)		
0x0800 + *n**0x100	XBAR Master General Purpose Control Register n (XBAR_MGPCRn)	*Section 17.3.3*

17.3.1 XBAR Priority Registers Slave (XBAR_PRS*n*)

The priority registers (XBAR_PRS*n*) set the priority of each master port on a per-slave-port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses.

Once the RO (Read Only) bit has been set in the XBAR Control Register (XBAR_CRSn), the Priority Register Slave (XBAR_PRSn) becomes read-only. Attempts to write on it will have no effect and will result in a bus-error response to the master initiating the write.

Note: Two available master ports cannot be programmed with the same priority level. Attempts to program two or more available masters with the same priority level will result in an error response and the priority will not be updated.

Refer to *Section 7.3.2: Crossbar switch* in *Chapter 7: Device configuration* for XBAR_PRS*n* priority values.

Offset: 0x0000 + *n**0x100 (*n* = 0 to 7) Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	M7			0	M6			0	M5			0	M4		
W																
Reset	0	0/1[1]	0/1[1]	0/1[1]	0	0/1[1]	0/1[1]	0/1[1]	0	0/1[1]	0/1[1]	0/1[1]	0	0/1[1]	0/1[1]	0/1[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M3			0	M2			0	M1			0	M0		
W																
Reset	0	0/1[1]	0/1[1]	0/1[1]	0	0/1[1]	0/1[1]	0/1[1]	0	0/1[1]	0/1[1]	0/1[1]	0	0/1[1]	0/1[1]	0/1[1]

1. Refer to PRSn register reset values in *Section 7.3.2: Crossbar switch* of *Chapter 7: Device configuration*.

Figure 138. Priority Registers Slave (XBAR_PRS*n*)

Table 215. XBAR_PRS*n* field descriptions

Field	Description
1:3 M7	Master 7 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
5:7 M6	Master 6 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
9:11 M5	Master 5 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
13:15 M4	Master 4 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.

Table 215. XBAR_PRS*n* field descriptions (continued)

Field	Description
17:19 M3	Master 3 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
21:23 M2	Master 2 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
25:27 M1	Master 1 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
29:31 M0	Master 0 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.

17.3.2 XBAR Control Register (XBAR_CRS*n*)

Offset: 0x0010 + *n**0x100 (*n* = 0 to 7) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	HRP	0	0	0	0	0	0	HPE7	HPE6	HPE5	HPE4	HPE3	HPE2	HPE1	HPE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ARB		0		PCTL		0	PARK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 139. XBAR Control Register (XBAR_CRS*n*)

Table 216. XBAR_CRS*n* field descriptions

Field	Description
0 RO	Read only Forces both of the slave port's registers to be read-only. After set, only a hardware reset clears it. 0 The slave port's registers are writable. 1 The slave port's registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HRP	Halt Request Priority Determines if a request to halt the crossbar is treated as the highest priority request, or the lowest priority request. 0 The halt request has the highest priority for arbitration on this slave port. 1 The halt request has the lowest initial priority for arbitration on this slave port.
8:15 HPE[7:0]	High Priority Enable (HPEx) Determines if Master x is able to temporarily elevate its request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master x are ignored and are treated the same as regular requests. 1 High Priority requests from Master x can temporarily elevate the priority level of the master's request to the slave.
22:23 ARB	Arbitration mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority 10 Reserved 11 Reserved

Table 216. XBAR_CRSn field descriptions (continued)

Field	Description
26:27 PCTL	Parking control Determines the slave port's parking control. The low-power park feature results in overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field. 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port. 10 When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state. 11 Reserved
29:31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. **Note:** Only select master ports that are actually present on the device. If not, undefined behavior may occur. Also, take care of the hardwired connections between Master and Flash slaves ports for applying meaningful parking configuration. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

17.3.3 Master General Purpose Control Register (XBAR_MGPCRn)

Offset: 0x0800 + *n**0x100 (*n* = 0 to 7) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0/1(1)	0/1(1)	0/1(1)

1. See MGPCRn register reset values in *Section 7.3.2: Crossbar switch* of *Chapter 7: Device configuration*.

Figure 140. Master General Purpose Control Register (XBAR_MGPCRn)

Table 217. XBAR_MGPCRn field descriptions

Field	Description
29:31 AULB	Arbitrate on Undefined Length Bursts These bits are used to select the arbitration policy during undefined length bursts by the master. These bits are initialized by hardware reset. 000 No arbitration will be allowed during undefined length burst 001 Arbitration will be allowed at any time during an undefined length burst 010 Arbitration will be allowed after 4 beats of an undefined length burst 011 Arbitration will be allowed after 8 beats of an undefined length burst 100 Arbitration will be allowed after 16 beats of an undefined length burst 101 Reserved 110 Reserved 111 Reserved

17.4 Functional description

This section provides the functional details of the crossbar switch.

17.4.1 General operation

When a master sends an access to the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. It is possible to make single-clock (zero wait state) accesses through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Since the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply enters a wait state.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
  - An outstanding request to one slave port that has a long response time and
  - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a fixed-length burst transfer it retains control of the slave port until that transfer completes. When a master has control of a given slave port, the crossbar returns all response information from the slave back to the requesting master.

The crossbar terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave buses). Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by the XBAR_CRS*n*[PARK]. This is done in an attempt to save the initial clock of arbitration delay that otherwise would be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low-power park mode in attempt to save power by using XBAR_CRS*n*[PCTL].

17.4.2 Register coherency

Since the content of the registers has a real-time effect on the operation of the crossbar, it is important to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave-port-related master accesses; instead, they track only with slave accesses.

17.4.3 Arbitration

The crossbar switch supports two arbitration schemes: a simple fixed-priority comparison algorithm and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

17.4.3.1 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the XBAR_PRS*n* (priority registers). If two masters request access to a slave port, the master with the higher priority in the selected priority register gains control over the slave port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master’s priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port performs an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master’s priority level is higher than that of the master that currently has control of the slave port, the new requesting master is granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed-length burst transfer or a locked transfer. In this case, the new requesting master must wait until the end of the burst transfer or locked transfer before it is granted control of the slave port.

If the new requesting master’s priority level is lower than the master that currently has control of the slave port, the new requesting master is forced to wait until the current master runs one of the following cycles:

- An IDLE cycle
- A non-IDLE cycle to a location other than the current slave port

17.4.3.2 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the physical master port number. This relative priority is compared to the ID (master port number) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far the ID of the requesting master is ahead of the ID of the last master.

After access is granted to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order 4, 5, and then 0.

Parking may continue to be used in a round-robin mode, but it does not affect the round-robin pointer unless the parked master actually performs a transfer. Hand-off occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

17.5 Initialization/application information

No initialization is required by or for the crossbar switch. Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. Settings and priorities should be programmed to achieve maximum system performance.

18 Crossbar Integrity Checker (XBIC)

18.1 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers. For forward signals (master to slave), it is done by verifying the integrity of the attribute information using an 8-bit Error Detection Code (EDC). The EDC detects any single- or double-bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. For feedback signals (slave to master), it is done by comparing the consistence of the signals during the AHB dataphase.There are three signals from slave to master, hready, hresp0, and hresp2. If any of the master signals is different from the slave signals during dataphase, the error will be reported in the Error Status Register.

During the address phase of the system bus pipelined protocol, the XBIC generates an 8-bit EDC checkbit value for every transfer, passing these checkbits through the crossbar to the targeted slave where it is compared against the 8-bit EDC checkbits calculated from the crossbar slave attributes. The EDC is calculated for 64 bits of attribute information: 32 bits of system bus attribute control plus 32 bits associated with the processor core’s decorated storage attribute. The resulting (72,64) code is a distance 4 encoding using a minimum odd weight H matrix definition. The EDC definition has been optimized for timing considerations with the system bus attribute signals exclusively using weight 3 code values and the decoration attribute using a mix of weight 3 and 5 codes.

During the data phase of the system bus pipelined protocol, the XBIC compares the three signals of hready, hresp0, hresp2 from entering the crossbar on the slave side to leaving the crossbar on the master side. Data phase is often more than one clock cycle due to wait cycles. All three signals are compared clock by clock during the whole data cycle.

18.2 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers
  - EDC (72,64) code protects against single and double bit errors
- Verification of feedback information for each data phase during crossbar transfer
  - hready, hresp0, and hresp2 are compared from entering the crossbar on the slave side to leaving the crossbar on the master side
- Error injection for testing
  - Programmable master and slave port specifiers
  - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
  - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable integrity feedback check enable on a per-master-port basis

18.3 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown in *Figure 141* and *Figure 142*.

Figure 141. XBIC system block diagram

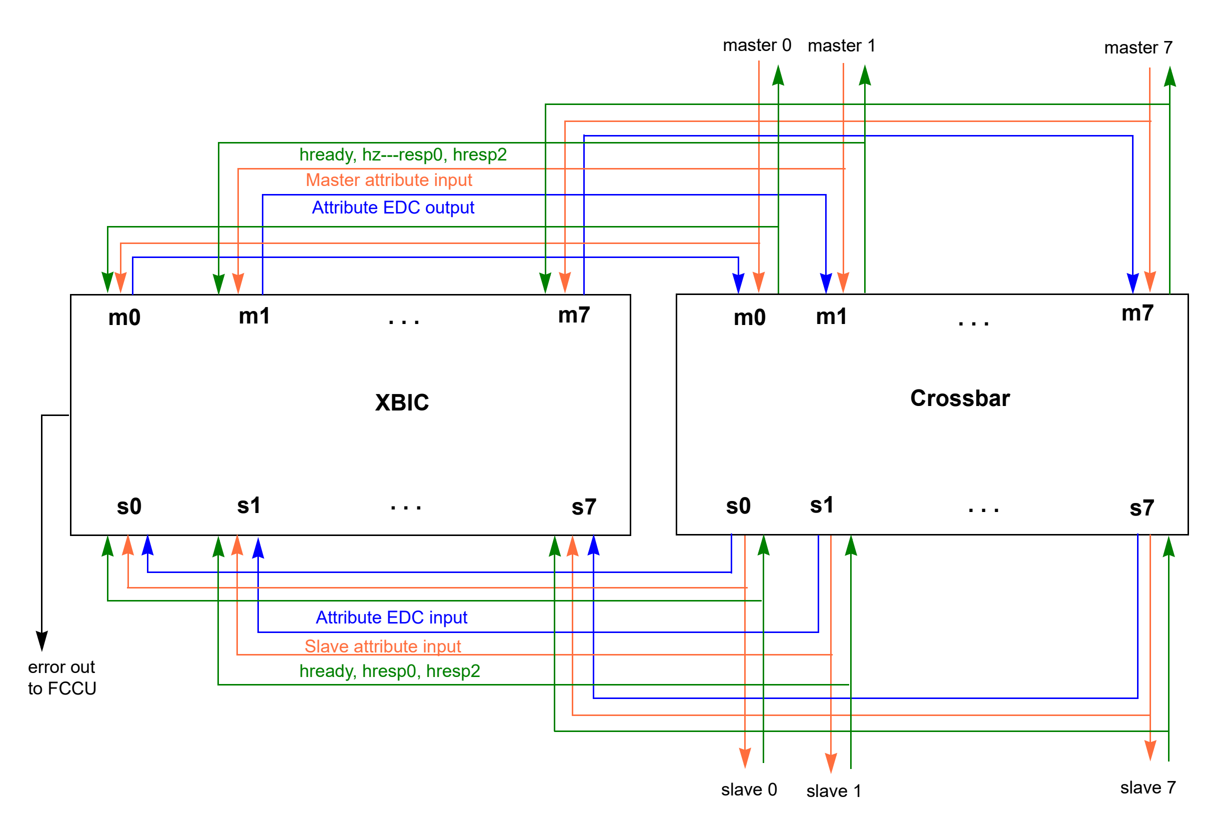
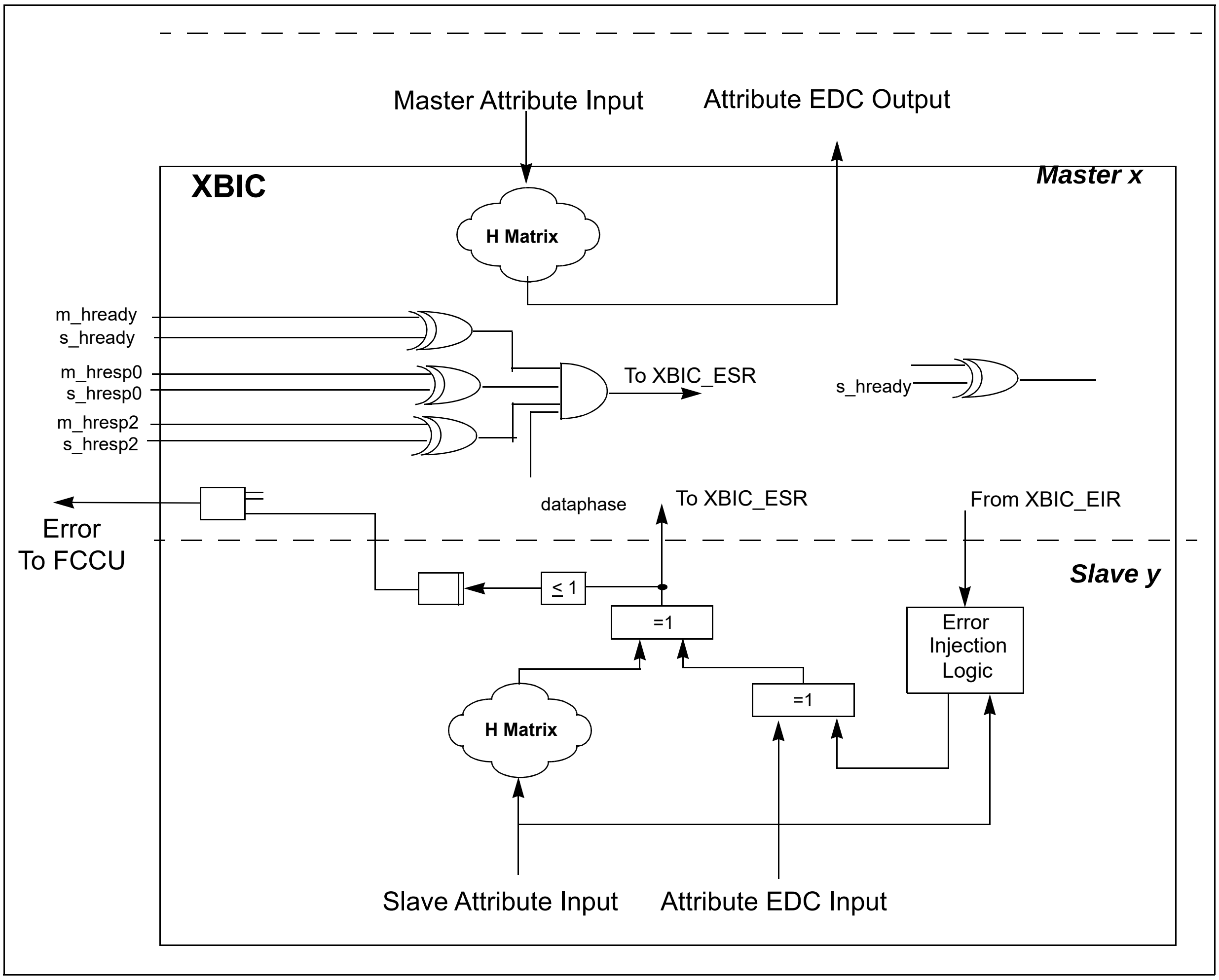


Figure 142. XBIC block diagram



18.4 External signal description

The XBIC has no external interface signals.

18.5 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register), or in user mode generate an error termination.

18.5.1 Memory map

The XBIC memory map is shown in *Table 218*.

Table 218. XBIC memory map

Address offset (hex)	Register description	Location
0x0000	XBIC Module Control Register (XBIC_MCR)	*Section 18.5.2.1*
0x0004	XBIC Error Injection Register (XBIC_EIR)	*Section 18.5.2.2*
0x0008	XBIC Error Status Register (XBIC_ESR)	*Section 18.5.2.3*
0x000C	XBIC Error Address Register (XBIC_EAR)	*Section 18.5.2.4*
0x0010–0x3FFF	Reserved	

18.5.2 Register descriptions

The following sections detail the individual registers within the XBIC programming model.

18.5.2.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Offset: 0x0000 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SE0	SE1	SE2	SE3	SE4	SE5	SE6	SE7	ME0	ME1	ME2	ME3	ME4	ME5	ME6	ME7(1)
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is writable but has no impact.

Figure 143. XBIC Module Control Register (XBIC_MCR)

Table 219. XBIC_MCR field descriptions

Field	Description
0 SE0	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 0 1 Attribute integrity checking enabled for slave port 0
1 SE1	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 1 1 Attribute integrity checking enabled for slave port 1
2 SE2	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 2 1 Attribute integrity checking enabled for slave port 2
3 SE3	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 3 1 Attribute integrity checking enabled for slave port 3

Table 219. XBIC_MCR field descriptions (continued)

Field	Description
4 SE4	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 4 1 Attribute integrity checking enabled for slave port 4
5 SE5	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 5 1 Attribute integrity checking enabled for slave port 5
6 SE6	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 6 1 Attribute integrity checking enabled for slave port 6
7 SE7	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 7 1 Attribute integrity checking enabled for slave port 7
8 ME0	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 0 1 Attribute integrity checking enabled for master port 0
9 ME1	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 1 1 Attribute integrity checking enabled for master port 1
10 ME2	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 2 1 Attribute integrity checking enabled for master port 2
11 ME3	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 3 1 Attribute integrity checking enabled for master port 3
12 ME4	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 4 1 Attribute integrity checking enabled for master port 4
13 ME5	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 5 1 Attribute integrity checking enabled for master port 5
14 ME6	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 6 1 Attribute integrity checking enabled for master port 6
15 ME7	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 7 1 Attribute integrity checking enabled for master port 7

18.5.2.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Offset: 0x0004 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 144. XBIC Error Injection Register (XBIC_EIR)

Table 220. XBIC_EIR field descriptions

Field	Description
0 EIE	Error Injection Enable 0 Error injection disabled 1 Error injection enabled
17:19 SLV	Target Slave Port (1) Error injection is enabled for the designated slave port. Other slave ports are unaffected.
20:23 MST	Target Master ID (2) Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
24:31 SYN	Syndrome This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

1. For the list of the target slave ports refer to the table “Crossbar switch slave assignments” in section “Crossbar switch” from “Device Configuration” chapter.
2. For the list of the target master IDs refer to the table “AHB Master ID Assignments” in section “Crossbar switch” from “Device Configuration” chapter.

18.5.2.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the *last* transfer. If an attribute integrity check error was detected, the slave port, the master port and the syndrome are captured in the SLV, MST and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are captured in the DPSE0*x* and DPME*x* fields.

This register is cleared only on reset.

Offset: 0x0008 Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	DPSE0	DPSE1	DPSE2	DPSE3	DPSE4	DPSE5	DPSE6	DPSE7	DPME0	DPME1	DPME2	DPME3	DPME4	DPME5	DPME6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 145. XBIC Error Status Register (XBIC_ESR)

Table 221. XBIC_ESR field descriptions

Field	Description
0 VLD	Error Status Valid 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
1 DPSE0	Data phase slave port error 0 No error on slave port 0 1 Data phase error on slave port 0
2 DPSE1	Data phase slave port error 0 No error on slave port 1 1 Data phase error on slave port 1
3 DPSE2	Data phase slave port error 0 No error on slave port 2 1 Data phase error on slave port 2
4 DPSE3	Data phase slave port error 0 No error on slave port 3 1 Data phase error on slave port 3
5 DPSE4	Data phase slave port error 0 No error on slave port 4 1 Data phase error on slave port 4
6 DPSE5	Data phase slave port error 0 No error on slave port 5 1 Data phase error on slave port 5
7 DPSE6	Data phase slave port error 0 No error on slave port 6 1 Data phase error on slave port 6
8 DPSE7	Data phase slave port error 0 No error on slave port 6 1 Data phase error on slave port 6

Table 221. XBIC_ESR field descriptions (continued)

Field	Description
9 DPME0	Data phase master port error 0 No error on master port 0 1 Data phase error on master port 0
10 DPME1	Data phase master port error 0 No error on master port 1 1 Data phase error on master port 1
11 DPME2	Data phase master port error 0 No error on master port 2 1 Data phase error on master port 2
12 DPME3	Data phase master port error 0 No error on master port 3 1 Data phase error on master port 3
13 DPME4	Data phase master port error 0 No error on master port 4 1 Data phase error on master port 4
14 DPME5	Data phase master port error 0 No error on master port 5 1 Data phase error on master port 5
15 DPME6	Data phase master port error 0 No error on master port 6 1 Data phase error on master port 6
17:19 SLV	Slave Port The slave port targeted by the last transfer with an error.
20:23 MST	Master ID The master ID value of the last transfer with an error.
24:31 SYN	Syndrome The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see *Table 223*.

18.5.2.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the last transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

Offset: 0x000C Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 146. Error Address Register (XBIC_EAR)

Table 222. XBIC_EAR Register field descriptions

Field	Description
0:31 ADDR	The address of the last transfer with an error.

18.6 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the attribute information for crossbar transfers using an 8-bit Error Detection Code (EDC). The EDC detects any single or double bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a per-slave-port basis. The EDC parity is generated by the XBIC for each master port during the system bus address phase pipeline stage, routed through the crossbar via a sideband signal and checked against the slave attribute EDC generated for each enabled slave port during the first system bus data phase cycle. Detection of an error does *not* generate a bus error. The XBIC attribute integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

The XBIC can be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. This approach allows the check logic to be verified without compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared.

Forwarding errors (master to slave) are detected during the AHB address phase. When a forwarding error is detected, due either to a hardware fault or error injection, information related to the error is captured and reported in the Error Status Register (XBIC_ESR) and Error Address Register (XBIC_EAR). The XBIC_ESR and XBIC_EAR registers contain information about the *last* transfer with a detected error and are only cleared on reset. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in *Table 223*. Any other syndrome indicates a multi-bit error.

XBIC also detects backward signal (slave to master) errors during AHB data phase. For the three slave driven signals (hready, hresp0, and hresp2), XBIC performs the comparison between the input of the slave port and output of the master port during the AHB data phase. When a mismatch occurs, the error will be captured. The current slave port will be registered in XBIC_ESR bit field DPSE*x*, and the current master port will be registered in XBIC_ESR bit field DSME*x*.

Table 223. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61

Table 223. Hexadecimal attribute single-bit error syndromes (continued)

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

The H matrix for the attribute EDC is shown in *Table 224*.

Table 224. XBIC H Matrix Definition

Checkbits [7:0]	Attribute Bit																															
	hwrite	htrans[0]	hsize[2]	hsize[1]	hsize[0]	hprot[5]	hprot[4]	hprot[3]	hprot[2]	hprot[1]	hprot[0]	hburst[2]	hburst[1]	hburst[0]	hmastlock	hunalign	hbstrb[7]	hbstrb[6]	hbstrb[5]	hbstrb[4]	hbsrtb[3]	hbsrtb[2]	hbstrb[1]	hbsrtb[0]	hmaster[3]	hmaster[2]	hmaster[1]	hmaster[0]	hslave[2]	hslave[1]	hslave[0]	hdecorated
7											*	*								*				*	*	*	*	*	*	*	*	*
6													*				*		*		*	*	*				*	*	*	*	*	*
5												*	*	*	*	*	*	*		*				*	*	*						*
4					*	*	*	*	*	*	*						*	*	*							*					*	
3		*	*	*				*	*	*				*	*	*				*			*							*		
2	*		*	*		*	*			*			*			*						*	*	*					*			
1	*	*		*	*		*		*						*			*	*		*				*			*				
0	*	*	*		*	*		*			*	*		*							*	*					*					

Checkbits [7:0]	hdecor[31]	hdecor[30]	hdecor[29]	hdecor[28]	hdecor[27]	hdecor[26]	hdecor[25]	hdecor[24]	hdecor[23]	hdecor[22]	hdecor[21]	hdecor[20]	hdecor[19]	hdecor[18]	hdecor[17]	hdecor[16]	hdecor[15]	hdecor[14]	hdecor[13]	hdecor[12]	hdecor[11]	hdecor[10]	hdecor[9]	hdecor[8]	hdecor[7]	hdecor[6]	hdecor[5]	hdecor[4]	hdecor[3]	hdecor[2]	hdecor[1]	hdecor[0]
7			*	*	*	*	*	*	*		*	*	*								*	*	*				*					
6		*	*							*				*		*		*	*	*	*	*	*		*		*				*	*
5	*	*													*		*			*	*	*	*	*		*	*	*	*	*		*
4											*	*	*		*			*	*				*	*	*	*	*	*	*	*		
3		*					*	*	*	*			*			*							*	*	*			*	*	*	*	*
2	*		*		*	*			*			*		*	*				*			*				*			*	*	*	*
1			*	*		*		*			*			*		*	*				*	*				*		*		*	*	*
0	*		*	*	*		*			*							*	*		*	*					*	*	*	*		*	

19 Peripheral Bridge (PBRIDGE)

19.1 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access a majority of peripherals on the device.

The peripheral bridge occupies a 64 MB portion of the address space (not all peripheral slots may be used. Refer to Device Configuration chapter or the Memory Map chapter for details on slot assignment).
The bridge includes separate clock enable inputs for each of the slots, to accommodate slower peripherals.

19.1.1 Features

Key features of the peripheral bridge are:

- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Each independently configurable peripheral includes a clock enable, which allows peripherals to operate at any speed less than the system clock rate
- Programming model that provides memory protection functionality.

19.1.2 General operation

The slave devices connected to the peripheral bridge are modules that contain readable/writable control and status registers. The system masters read and write these registers through the peripheral bridge. The peripheral bridge generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the peripherals. The peripheral bridge captures read data from the peripheral interface and drives it to the crossbar switch.

Each peripheral is typically allocated one 16-KB block (or slot) of the memory map. Usually, the peripheral bridge uses the size of the addressed peripheral to perform proper data byte lane routing only; no bus decomposition (dynamic bus sizing) is performed. However, there are the following exceptions:

- Aligned 64-bit reads are permitted to 32-bit slots.
- Aligned 64-bit writes may be targeted to 32-bit slots.
- Aligned or misaligned 64-bit instruction fetches may also be performed to 32-bit slots.

All other 64-bit accesses result in an error response.

19.2 Memory map and register definition

Table 225. Peripheral bridge memory map

Offset address	Register	Location
0x000	Master Privilege Register A (MPRA)	Section 19.2.1
0x004	Master Privilege Register B (MPRB)	Section 19.2.2
0x008–0x1F	Reserved[2]	

Table 225. Peripheral bridge memory map (continued)

Offset address	Register	Location
0x020–0x2F	Reserved(1)	
0x030–0x3F	Reserved(2)	
0x040–0xBF	Reserved(1)	
0x0C0–0xDF	Reserved(2)	
0x0E0–0xFF	Reserved(1)	
0x100	Peripheral Access Control Register A (PACRA)	Section 19.2.3
0x104	Peripheral Access Control Register B (PACRB)	Section 19.2.3
0x108	Peripheral Access Control Register C (PACRC)	Section 19.2.3
0x10C	Peripheral Access Control Register D (PACRD)	Section 19.2.3
0x110	Peripheral Access Control Register E (PACRE)	Section 19.2.3
0x114	Peripheral Access Control Register F (PACRF)	Section 19.2.3
0x118	Peripheral Access Control Register G (PACRG)	Section 19.2.3
0x11C	Peripheral Access Control Register H (PACRH)	Section 19.2.3
0x120–0x12F	Reserved(1)	
0x130–0x13F	Reserved(2)	
0x140	Off-platform Peripheral Access Control Register A (OPACRA)	Section 19.2.4
0x144	Off-platform Peripheral Access Control Register B (OPACRB)	Section 19.2.4
0x148	Off-platform Peripheral Access Control Register C (OPACRC)	Section 19.2.4
0x14C	Off-platform Peripheral Access Control Register D (OPACRD)	Section 19.2.4
0x150	Off-platform Peripheral Access Control Register E (OPACRE)	Section 19.2.4
0x154	Off-platform Peripheral Access Control Register F (OPACRF)	Section 19.2.4
0x158	Off-platform Peripheral Access Control Register G (OPACRG)	Section 19.2.4
0x15C	Off-platform Peripheral Access Control Register H (OPACRH)	Section 19.2.4
0x160	Off-platform Peripheral Access Control Register I (OPACRI)	Section 19.2.4
0x164	Off-platform Peripheral Access Control Register J (OPACRJ)	Section 19.2.4
0x168	Off-platform Peripheral Access Control Register K (OPACRK)	Section 19.2.4
0x16C	Off-platform Peripheral Access Control Register L (OPACRL)	Section 19.2.4
0x170	Off-platform Peripheral Access Control Register M (OPACRM)	Section 19.2.4
0x174	Off-platform Peripheral Access Control Register N (OPACRN)	Section 19.2.4
0x178	Off-platform Peripheral Access Control Register O (OPACRO)	Section 19.2.4
0x17C	Off-platform Peripheral Access Control Register P (OPACRP)	Section 19.2.4
0x180	Off-platform Peripheral Access Control Register Q (OPACRQ)	Section 19.2.4
0x184	Off-platform Peripheral Access Control Register R (OPACRR)	Section 19.2.4
0x188	Off-platform Peripheral Access Control Register S (OPACRS)	Section 19.2.4

Table 225. Peripheral bridge memory map (continued)

Offset address	Register	Location
0x18C	Off-platform Peripheral Access Control Register T (OPACRT)	*Section 19.2.4*
0x190	Off-platform Peripheral Access Control Register U (OPACRU)	*Section 19.2.4*
0x194	Off-platform Peripheral Access Control Register V (OPACRV)	*Section 19.2.4*
0x198	Off-platform Peripheral Access Control Register W (OPACRW)	*Section 19.2.4*
0x19C	Off-platform Peripheral Access Control Register X (OPACRX)	*Section 19.2.4*
0x1A0	Off-platform Peripheral Access Control Register Y (OPACRY)	*Section 19.2.4*
0x1A4	Off-platform Peripheral Access Control Register Z (OPACRZ)	*Section 19.2.4*
0x1A8	Off-platform Peripheral Access Control Register AA (OPACRAA)	*Section 19.2.4*
0x1AC	Off-platform Peripheral Access Control Register AB (OPACRAB)	*Section 19.2.4*
0x1B0	Off-platform Peripheral Access Control Register AC (OPACRAC)	*Section 19.2.4*
0x1B4	Off-platform Peripheral Access Control Register AD (OPACRAD)	*Section 19.2.4*
0x1B8	Off-platform Peripheral Access Control Register AE (OPACRAE)	*Section 19.2.4*
0x1BC	Off-platform Peripheral Access Control Register AF (OPACRAF)	*Section 19.2.4*

1. The following address spaces when programmed, will not terminate in an error. Software should not program these address spaces.
2. Access to this address space will terminate in an error.

19.2.1 Master Privilege Register A (MPRA)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the device to the various peripherals. The register provides one field per bus master. Each master is assigned depending on its logical master number. Refer to Device Configuration chapter for details about the master assignments to these registers.

The MPROT*n* field is defined as shown in *Figure 147*.

	0	1	2	3
R	0	MTR	MTW	MPL
W				

Figure 147. MPROT*n* fields

Table 226. MPROT*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 MTR	Master trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.

Table 226. MPROTn field descriptions (continued)

Field	Description
2 MTW	Master trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL	Master privilege level Determines how the privilege level of the master is determined. 0 Accesses from this master are forced to user mode. 1 Accesses from this master are not forced to user mode.

Address: 0x000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MPROT0				MPROT1				MPROT2				MPROT3				MPROT4				MPROT5				MPROT6				MPROT7			
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

Figure 148. Master Privilege Register A (MPRA)

Table 227. MPRA field descriptions

Field	Description
0:3 MPROT0	Access privilege level associated with bus master 0
4:7 MPROT1	Access privilege level associated with bus master 1
8:11 MPROT2	Access privilege level associated with bus master 2
12:15 MPROT3	Access privilege level associated with bus master 3
16:19 MPROT4	Access privilege level associated with bus master 4
20:23 MPROT5	Access privilege level associated with bus master 5
24:27 MPROT6	Access privilege level associated with bus master 6
28:31 MPROT7	Access privilege level associated with bus master 7

19.2.2 Master Privilege Register B (MPRB)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the device to the various peripherals. The register provides one field per bus master. Each master is assigned depending on its logical master number. Refer to Device Configuration chapter for details about the master assignments to these registers.

Address: 0x004 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MPROT8				MPROT9				MPROT10				MPROT11				MPROT12				MPROT13				MPROT14				MPROT15			
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

Figure 149. Master Privilege Register B (MPRB)

Table 228. MPRB field descriptions

Field	Description
0:3 MPROT8	Access privilege level associated with bus master 8
4:7 MPROT9	Access privilege level associated with bus master 9
8:11 MPROT10	Access privilege level associated with bus master 10
12:15 MPROT11	Access privilege level associated with bus master 11
16:19 MPROT12	Access privilege level associated with bus master 12
20:23 MPROT13	Access privilege level associated with bus master 13
24:27 MPROT14	Access privilege level associated with bus master 14
28:31 MPROT15	Access privilege level associated with bus master 15

19.2.3 Peripheral Access Control Register (PACR*x*)

Each of the on-platform peripherals has a four-bit PACR*n* field which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACR*x* register.

The peripheral assignments to each PACR field are defined by the memory map slot to which the peripherals are assigned. Refer to Memory Map chapter for the assignments of the device. If a peripheral is absent, the corresponding PACR field is not implemented. Reads to the location return zeros, and writes are ignored.

The PBRIDGE registers themselves are mapped into the PACR0 address space and can only be accessed in supervisor mode by trusted bus masters.

Table 229 shows the top-level structure of the PACRs.

Table 229. PACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x0104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x0108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x010C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x0110	PACRE	PACR32	PACR33	PACR34	PACR35	PACR36	PACR37	PACR38	PACR39
0x0114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x0118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x011C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

Each PACR*n* field is defined as shown in *Figure 150*.

	0	1	2	3
R	0	SP	WP	TP
W				

Figure 150. PACR*n* field

Table 230. PACR*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT*n*[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

Address: 0x100–0x11C (PACRA–PACRH) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	PACR*a*				PACRb				PACRc				PACRd				PACRe				PACRf				PACRg				PACRh			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Note: The reset value of PACR0 is 0101b.

Figure 151. Peripheral Access Control Registers (PACR*x*)

Table 231. PACR*x* field descriptions

Field	Description
0:3 PACRa	Access level associated with module a
4:7 PACRb	Access level associated with module b
8:11 PACRc	Access level associated with module c
12:15 PACRd	Access level associated with module d
16:19 PACRe	Access level associated with module e
20:23 PACRf	Access level associated with module f
24:27 PACRg	Access level associated with module g
28:31 PACRh	Access level associated with module h

19.2.4 Off-Platform Peripheral Access Control Registers (OPACR*x*)

Each of the off-platform peripherals has a four-bit OPACR*n* field which defines the access levels supported by the given module. Eight OPACRs are grouped together to form a 32-bit OPACR*x* register.

The peripheral assignments to each OPACR field are defined by the memory map slot to which the peripherals are assigned. Refer to Memory Map chapter for the assignments of the device. If a peripheral is absent, the corresponding OPACR is not implemented. Reads to the location return zeros and writes are ignored.

Table 232 shows the top-level structure of the OPACR registers.

Table 232. OPACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0140	OPACRA	OPACR 0	OPACR 1	OPACR 2	OPACR 3	OPACR 4	OPACR 5	OPACR 6	OPACR 7
0x0144	OPACRB	OPACR 8	OPACR 9	OPACR 10	OPACR 11	OPACR 12	OPACR 13	OPACR 14	OPACR 15
0x0148	OPACRC	OPACR 16	OPACR 17	OPACR 18	OPACR 19	OPACR 20	OPACR 21	OPACR 22	OPACR 23
0x014C	OPACRD	OPACR 24	OPACR 25	OPACR 26	OPACR 27	OPACR 28	OPACR 29	OPACR 30	OPACR 31
0x0150	OPACRE	OPACR 32	OPACR 33	OPACR 34	OPACR 35	OPACR 36	OPACR 37	OPACR 38	OPACR 39
0x0154	OPACRF	OPACR 40	OPACR 41	OPACR 42	OPACR 43	OPACR 44	OPACR 45	OPACR 46	OPACR 47
0x0158	OPACRG	OPACR 48	OPACR 49	OPACR 50	OPACR 51	OPACR 52	OPACR 53	OPACR 54	OPACR 55
0x015C	OPACRH	OPACR 56	OPACR 57	OPACR 58	OPACR 59	OPACR 60	OPACR 61	OPACR 62	OPACR 63
0x0160	OPACRI	OPACR 64	OPACR 65	OPACR 66	OPACR 67	OPACR 68	OPACR 69	OPACR 70	OPACR 71
0x0164	OPACRJ	OPACR 72	OPACR 73	OPACR 74	OPACR 75	OPACR 76	OPACR 77	OPACR 78	OPACR 79
0x0168	OPACRK	OPACR 80	OPACR 81	OPACR 82	OPACR 83	OPACR 84	OPACR 85	OPACR 86	OPACR 87
0x016C	OPACRL	OPACR 88	OPACR 89	OPACR 90	OPACR 91	OPACR 92	OPACR 93	OPACR 94	OPACR 95
0x0170	OPACRM	OPACR 96	OPACR 97	OPACR 98	OPACR 99	OPACR 100	OPACR 101	OPACR 102	OPACR 103
0x0174	OPACRN	OPACR 104	OPACR 105	OPACR 106	OPACR 107	OPACR 108	OPACR 109	OPACR 110	OPACR 111
0x0178	OPACRO	OPACR 112	OPACR 113	OPACR 114	OPACR 115	OPACR 116	OPACR 117	OPACR 118	OPACR 119
0x017C	OPACRP	OPACR 120	OPACR 121	OPACR 122	OPACR 123	OPACR 124	OPACR 125	OPACR 126	OPACR 127
0x0180	OPACRQ	OPACR 128	OPACR 129	OPACR 130	OPACR 131	OPACR 132	OPACR 133	OPACR 134	OPACR 135
0x0184	OPACRR	OPACR 136	OPACR 137	OPACR 138	OPACR 139	OPACR 140	OPACR 141	OPACR 142	OPACR 143
0x0188	OPACRS	OPACR 144	OPACR 145	OPACR 146	OPACR 147	OPACR 148	OPACR 149	OPACR 150	OPACR 151

Table 232. OPACR memory map (continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x018C	OPACRT	OPACR 152	OPACR 153	OPACR 154	OPACR 155	OPACR 156	OPACR 157	OPACR 158	OPACR 159
0x0190	OPACRU	OPACR 160	OPACR 161	OPACR 162	OPACR 163	OPACR 164	OPACR 165	OPACR 166	OPACR 167
0x0194	OPACRV	OPACR 168	OPACR 169	OPACR 170	OPACR 171	OPACR 172	OPACR 173	OPACR 174	OPACR 175
0x0198	OPACRW	OPACR 176	OPACR 177	OPACR 178	OPACR 179	OPACR 180	OPACR 181	OPACR 182	OPACR 183
0x019C	OPACRX	OPACR 184	OPACR 185	OPACR 186	OPACR 187	OPACR 188	OPACR 189	OPACR 190	OPACR 191
0x01A0	OPACRY	OPACR 192	OPACR 193	OPACR 194	OPACR 195	OPACR 196	OPACR 197	OPACR 198	OPACR 199
0x01A4	OPACRZ	OPACR 200	OPACR 201	OPACR 202	OPACR 203	OPACR 204	OPACR 205	OPACR 206	OPACR 207
0x01A8	OPACRAA	OPACR 208	OPACR 209	OPACR 210	OPACR 211	OPACR 212	OPACR 213	OPACR 214	OPACR 215
0x01AC	OPACRAB	OPACR 216	OPACR 217	OPACR 218	OPACR 219	OPACR 220	OPACR 221	OPACR 222	OPACR 223
0x01B0	OPACRAC	OPACR 224	OPACR 225	OPACR 226	OPACR 227	OPACR 228	OPACR 229	OPACR 230	OPACR 231
0x01B4	OPACRAD	OPACR 232	OPACR 233	OPACR 234	OPACR 235	OPACR 236	OPACR 237	OPACR 238	OPACR 239
0x01B8	OPACRAE	OPACR 240	OPACR 241	OPACR 242	OPACR 243	OPACR 244	OPACR 245	OPACR 246	OPACR 247
0x01BC	OPACRAF	OPACR 248	OPACR 249	OPACR 250	OPACR 251	OPACR 252	OPACR 253	OPACR 254	OPACR 255

The OPACR*n* field is defined as in *Figure 152*.

	0	1	2	3
R	0	SP	WP	TP
W				

Figure 152. OPACR*n* fields

Table 233. OPACR*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT*n*[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

Address: 0x140–0x1BC (OPACRA–OPACRAF) Access: Supervisor read/write

	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
R W	OPACR*a*	OPACRb	OPACRc	OPACRd	OPACRe	OPACRf	OPACRg	OPACRh
Reset	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0

Figure 153. Off-Platform Peripheral Access Control Registers (OPACR*x*)

Table 234. OPACR*x* field descriptions

Field	Description
0:3 OPACRa	Access level associated with module a
4:7 OPACRb	Access level associated with module b
8:11 OPACRc	Access level associated with module c
12:15 OPACRd	Access level associated with module d
16:19 OPACRe	Access level associated with module e
20:23 OPACRf	Access level associated with module f

Table 234. OPACR*x* field descriptions (continued)

Field	Description
24:27 OPACRg	Access level associated with module g
28:31 OPACRh	Access level associated with module h

19.3 Functional description

The peripheral bridge serves as an interface between the crossbar switch and the slave peripheral bus. It functions as a protocol translator. Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

Accesses which fall within the address space of the peripheral bridge are decoded to provide individual module selects for peripheral devices on the slave bus interface.

19.3.1 Access support

Aligned and misaligned 32-bit and 16-bit accesses, as well as byte accesses, are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. The peripheral bridge performs two slave peripheral bus transfers for allowed 64-bit transactions, to 32-bit sized peripheral slots only. All other accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted which is larger (in size) than the targeted port, an error response is generated without a peripheral access.

20 System Memory Protection Unit (SMPU)

20.1 Overview

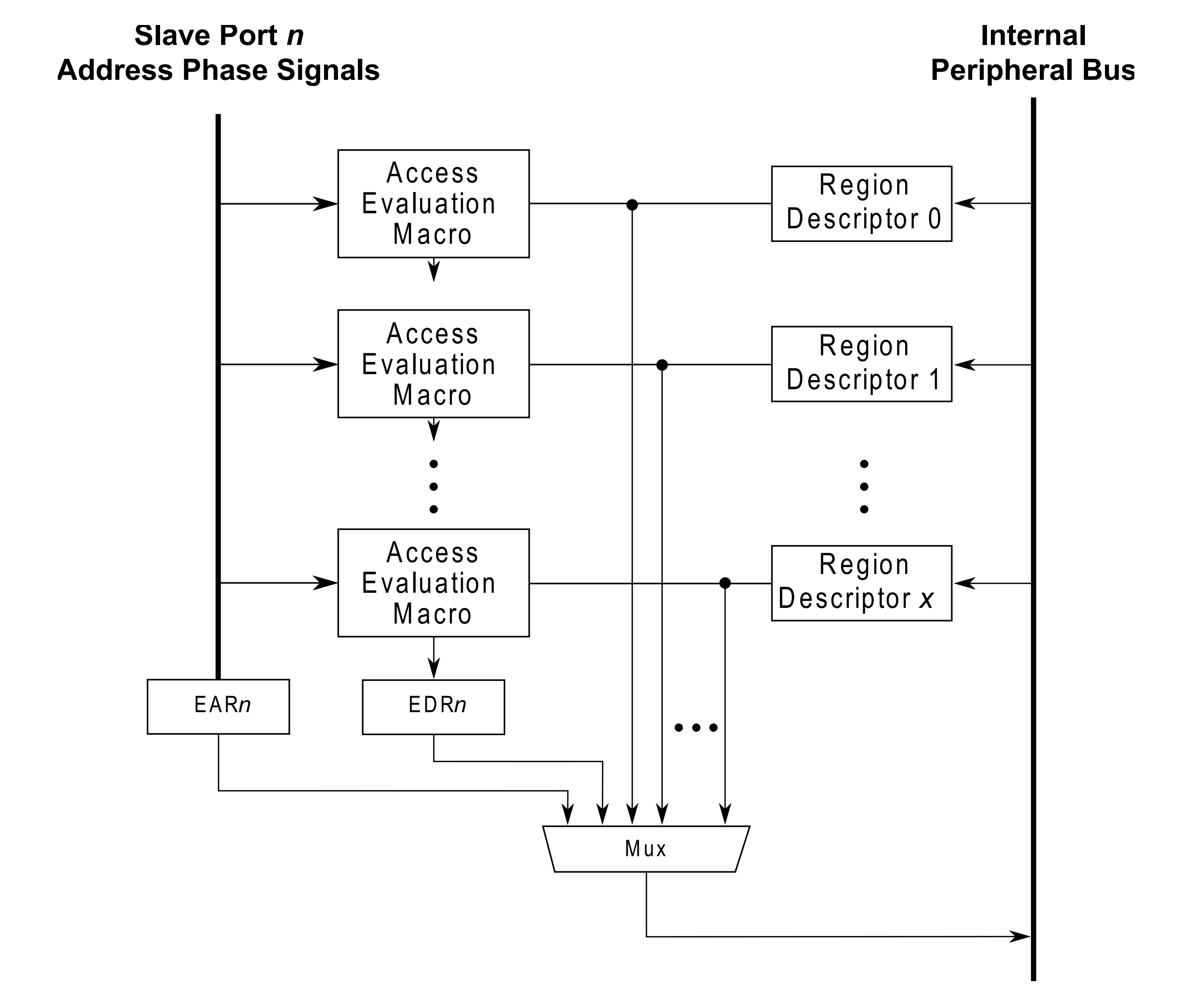
The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors and evaluates system bus transactions using preprogrammed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

20.2 Block diagram

A simplified block diagram of the SMPU module is shown in *Figure 154*. The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers are in the middle, and the peripheral bus interface is on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and detect protection violations.

For details of the access evaluation macro, refer to *Section 20.5.1: Access evaluation macro*.

Figure 154. SMPU block diagram



20.3 Features

The SMPU feature set includes:

- Supports up to 24 program-visible 128-bit region descriptors, accessible as four 32-bit words each. (Specific module instances may support fewer than 24)
  - Each region descriptor defines an arbitrarily sized space, aligned anywhere in memory
    Region sizes can vary from a minimum of 1 byte to a maximum of (4 GB minus 1 byte)
  - Read/write access control permissions defined in region descriptor
  - Cache-inhibit attribute indicator per region descriptor
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
  - Priority given to granting permission over denying access for overlapping region descriptors
- Supports as many as eight crossbar slave ports
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enable/disable control bit

20.4 Memory map and register definition

20.4.1 Memory map

The programming model is partitioned into three groups: control registers, error capture registers, and the data structure containing the region descriptors.

The programming model can only be referenced using 32-bit accesses. Attempted references using different access sizes, to undefined (reserved) addresses, or with a non-supported access type (a write to a read-only register, or a read of a write-only register) generate an error termination.

The programming model can be accessed only in supervisor mode.

Note: Refer to the device configuration details for any chip-specific register information for this module. For example, a specific instance of a module may support only up to 8 region descriptors.

Table 235. SMPU memory map

Address offset	Register name	Section
0x0000	CESR0	Section 20.4.2.1
0x0004	CESR1	Section 20.4.2.2
(n=0 to 15)		
0x8*n+0x0100	EARn	Section 20.4.2.3
0x8*n+0x0104	EDRn	Section 20.4.2.4
(n=0 to 23)		
0x10*n+0x0400	RGDn_WORD0	Section 20.4.2.5

Table 235. SMPU memory map (continued)

Address offset	Register name	Section
0x10*n+0x0404	RGDn_WORD1	*Section 20.4.2.6*
0x10*n+0x0408	RGDn_WORD2_FMT0	*Section 20.4.2.7*
0x10*n+0x040C	RGDn_WORD3	*Section 20.4.2.8*

20.4.2 Register descriptions

20.4.2.1 Control/Error Status Register 0 (CESR0)

Address: 0x0000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MERR															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	0	0	0	0	0	0	0	0	0	0	HRL			GVLD
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 155. Control/Error Status Register 0 (CESR0)

Table 236. CESR0 field descriptions

Field	Description
0:15 MERR	Master *n* error, where the bus master number matches the bit number Indicates a captured error in EAR*n* and EDR*n*. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing ‘1’ to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master *n.* 1 An error has occurred for bus master *n.*
16	Reserved This read-only bit is reserved and always has the value ‘1’.
28:30 HRL	Hardware revision level Specifies the SMPU’s hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled.

20.4.2.2 Control/Error Status Register 1 (CESR1)

Address: 0x0004 Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEOVR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	0	0	0	0	0	0	0	0	0	0	NRGD			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 156. Control/Error Status Register 1 (CESR1)

Table 237. CESR1 field descriptions

Field	Description
0:15 MEOVR	Master *n* error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master *n* has occurred before the previous error was processed. The details of the first error are recorded in the EAR*n* and EDR*n* registers, and no information on subsequent errors is recorded until the associated CESR0.MERR flag is cleared. 0 No error overrun condition has been detected for bus master *n.* 1 An error overrun condition has been detected for bus master *n.*
16	Reserved This read-only bit is reserved and always has the value ‘1’.
28:31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

20.4.2.3 Error Address Register, Bus Master *n* (EAR*n*)

When the SMPU detects an access error on bus master *n*, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0.MERR is set. Additional information about the faulting access is captured in the corresponding EDR*n* at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1.MEOVR field until the original error is processed.

Address: 0x0100 + *n**0x8 (*n* = 0 to 15) Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 157. Error Address Register, Bus Master *n* (EAR*n*)

Table 238. EAR*n* field descriptions

Field	Description
0:31 EADDR	Error address Indicates the reference address from bus master *n* that generated the access error.

20.4.2.4 Error Detail Register, Bus Master *n* (EDR*n*)

When the SMPU detects an access error associated with bus master *n*, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0.MERR is set. Information on the faulting address is captured in the corresponding EAR*n* register at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1.MEOVR field until the original error is processed.

Address: 0x0104 + *n**0x8 (*n* = 0 to 15) Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EACD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EACD								1	EATTR		ERW	EMN			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 158. Error Detail Register, Bus Master *n* (EDR*n*)

Table 239. EDRn field descriptions

Field	Description
0:23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDR*n* contains a captured error and EACD is all zeroes, an access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24	Reserved This read-only bit is reserved and always has the value ‘1’.
25:26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
28:31 EMN	Error master number Indicates the logical bus master number of the faulting reference. This field is used to determine the bus master that generated the access error.

20.4.2.5 Region Descriptor *n*, Word 0 (RGD*n*_WORD0)

Address: 0x0400 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 159. Region Descriptor *n*, Word 0 (RGD*n*_WORD0)

Table 240. RGDn_WORD0 field descriptions

Field	Description
0:31 SRTADDR	Start address Defines the byte start address of the memory region.

20.4.2.6 Region Descriptor *n*, Word 1 (RGD*n*_WORD1)

Address: 0x0404 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 160. Region Descriptor *n*, Word 1 (RGD*n*_WORD1)

Table 241. RGD*n*_WORD1 field descriptions

Field	Description
0:31 ENDADDR	End address Defines the byte end address of the memory region. **Note:** The SMPU does not verify that ENDADDR ≥ SRTADDR.

20.4.2.7 Region Descriptor *n*, Word 2 Format 0 (RGD*n*_WORD2_FMT0)

RGD_WORD2 has two formats as determined by the RGD_WORD3.FMT field.

Note: RGD_WORD2_FMT0 applies when RGD_WORD3[FMT] = 0.

RGD_WORD2_FMT0 defines the access control rights of the memory region on a per master basis. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags per logical bus master:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) instruction.

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Address: 0x0408 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 161. Region Descriptor *n*, Word 2 Format 0 (RGD*n*_WORD2_FMT0)

Table 242. RGD*n*_WORD2_FMT0 field descriptions

Field	Description
0:1 M0P	Bus master 0 permissions
2:3 M1P	Bus master 1 permissions
4:5 M2P	Bus master 2 permissions
6:7 M3P	Bus master 3 permissions
8:9 M4P	Bus master 4 permissions
10:11 M5P	Bus master 5 permissions
12:13 M6P	Bus master 6 permissions
14:15 M7P	Bus master 7 permissions
16:17 M8P	Bus master 8 permissions
18:19 M9P	Bus master 9 permissions
20:21 M10P	Bus master 10 permissions
22:23 M11P	Bus master 11 permissions
24:25 M12P	Bus master 12 permissions
26:27 M13P	Bus master 13 permissions

Table 242. RGD*n*_WORD2_FMT0 field descriptions (continued)

Field	Description
28:29 M14P	Bus master 14 permissions
30:31 M15P	Bus master 15 permissions

20.4.2.8 Region Descriptor *n*, Word 3 (RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, the format select (FMT), plus other configuration bits.

Address: 0x040C + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	FMT	RO	0	CI	VLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 162. Region Descriptor n, Word 3 (RGDn_WORD3)

Table 243. RGD*n*_WORD3 field descriptions

Field	Description
27 FMT	Region Descriptor Format This bit selects the configuration format (FMT0 or FMT1) for this region descriptor. **Note:** A specific module instance of the SMPU may support only the FMT0 format. If so, the FMT field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies. 0 Use format 0 (RGD_WORD2_FMT0) 1 Use format 1 (RGD_WORD2_FMT1)
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **Note:** Setting RO in an RGD locks all four words of the RGD until a system reset; the valid bit of the RGD and the global valid bit have no effect. 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.

Table 243. RGDn_WORD3 field descriptions (continued)

Field	Description
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **Note:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size. 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals that the region descriptor is valid. Any write to RGD*n*_WORD0–2 clears this bit. 0 Region descriptor is invalid. 1 Region descriptor is valid.

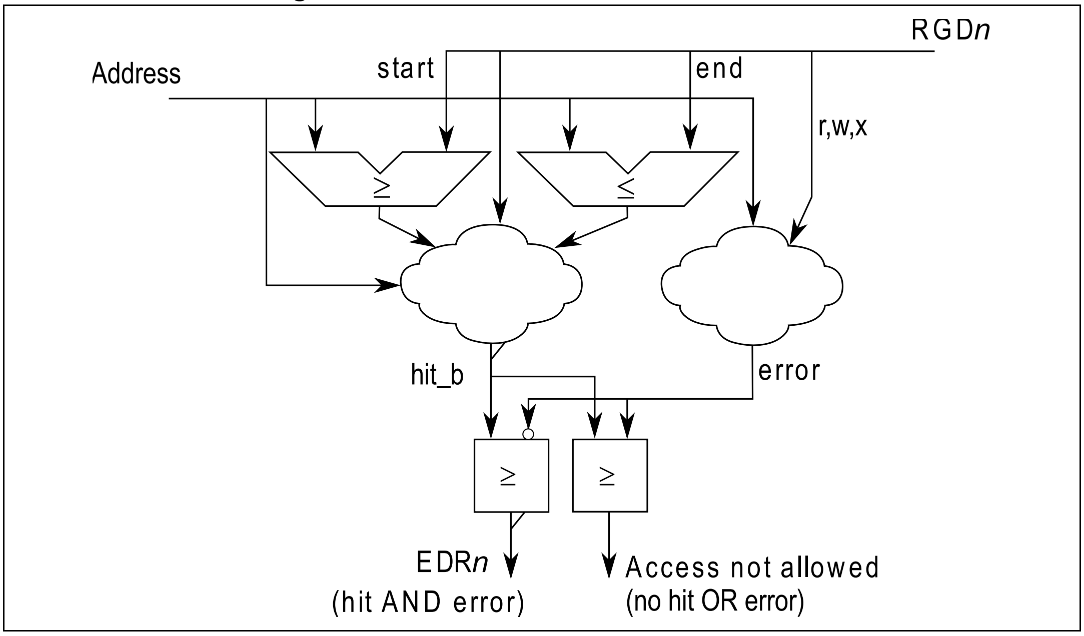
20.5 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

20.5.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in *Figure 163*, the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGD*n*), then performs two major functions: region hit determination and detection of an access protection violation. *Figure 163* shows a functional block diagram.

Figure 163. SMPU access evaluation macro



20.5.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. The boolean equation for this portion of the hit determination is:

```
region_hit = ((addr[0:31] >= RGDn_Word0[SRTADDR]) &
      (addr[0:31] <= RGDn_Word1[ENDADDR])) &
RGDn_Word3[VLD]
```

Where addr is the current reference address, RGDn_Word0.SRTADDR and RGDn_Word1.ENDADDR are the start and end addresses, and RGDn_Word3.VLD is the valid bit.

Note: The SMPU does not verify that ENDADDR ≥ SRTADDR.

Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

20.5.1.2 Privilege violation determination

While the access evaluation macro is determining the region hit, the logic is also evaluating whether the current access is allowed by the permissions defined in the region descriptor. Using the master signal, a set of effective permissions is generated from the relevant fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions, using the specification shown in *Table 244*.

Table 244. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

20.5.2 Putting it all together and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, refer to *Section 20.7*.

20.6 Initialization information

At system startup, load the required number of region descriptors, including setting RGD*n*_Word3.VLD. Setting CESR0.GVLD enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0.GVLD = 0). Next, load the appropriate region descriptors (RGD*n*), including the setting of the RGDn_Word3.VLD bits. Finally, set CESR0.GVLD to enable the entire module.

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

20.7 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities:

- Creating a new memory region: Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Modifying a region descriptor: Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Removing a region descriptor: Clearing RGD*n*_Word3.VLD deletes an existing region descriptor.
- Accessing the SMPU: Allocate a region descriptor to restrict SMPU access to supervisor mode from a specific master.
- Detecting an access error: The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0.MERR signals. The error registers contain the captured fault data.
- Overlapping region descriptors: Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

21 Intelligent AHB Gasket (IAHBG)

21.1 Introduction

This section details the intelligent bridging gasket between the fast and slow clock domains. The intelligent operation of the gasket initiates pending reads early and optimizes bursts to recover some of the performance that is lost on a registered interface between different clock domains. Pending reads begin even if there is not an hready to the master. Non-bursted writes only begin when the slave is IDLE, pending writes wait until IDLE.

The IAHBG complies with the AHBV6lite protocol, including an extension to support masters that do address retraction during the error termination sequence or abort a burst read due to an error termination or another non-error termination like cache inhibit termination.

Table 245 lists the available operation modes.

Table 245. Operation modes available

Operating mode (Master:Slave)	Comments
0:0	Flowthru (no register wall)
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

21.2 Timing modes

21.2.1 1:1

In the 1:1 timing mode the master is running at the same clock frequency as the slave. The pending read can either be a single or burst transaction.

21.2.2 2:1

In the 2:1 timing mode the master is running at twice the clock frequency of the slave. The 4:1 mode is very similar to the 2:1 mode. The pending read can either be a single or burst transaction.

21.2.3 1:2

In the 1:2 timing mode, the slave is running at twice the clock frequency of the master.The 1:4 mode is very similar to the 1:2 mode. The pending read can either be a single or burst transaction.

Because the slave is operating at a faster clock frequency than the master, the transactions from the master must be stored into a queue of data and response attribute registers. At a certain point during the master burst transactions, the queue is either emptied into the master for a burst read or it is emptied into the slave for a burst write.

22 Interrupt Controller (INTC)

22.1 Introduction

Note: For the chip-specific implementation details of this module's instances, refer to the Section 7.3.5: Interrupt controller (INTC) configuration.

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to the processor (PRC)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 64 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

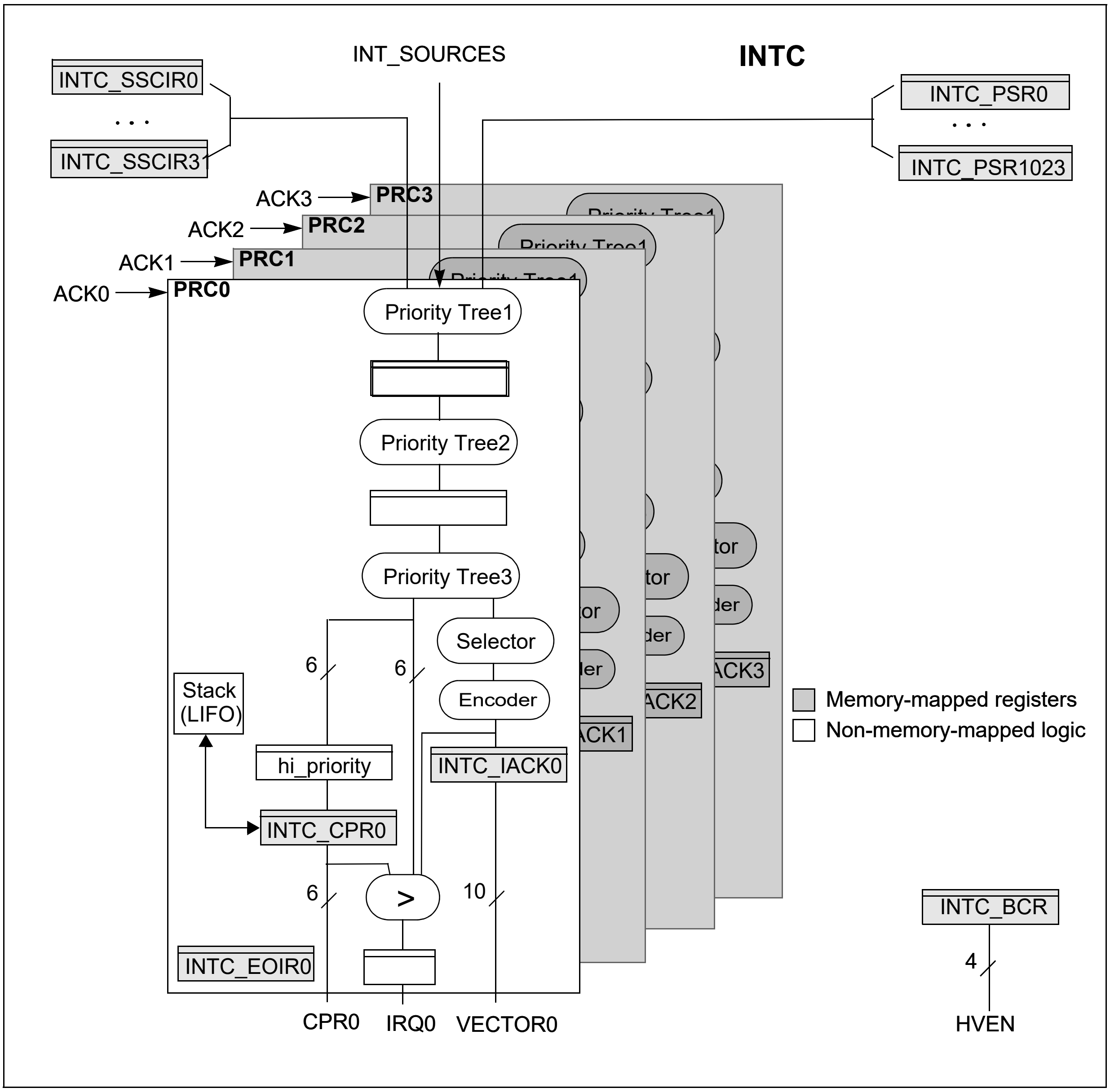
When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

The software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

22.2 Block diagram

The block diagram for the INTC is shown in *Figure 164*.

Figure 164. INTC block diagram



22.3 Features

- Each peripheral interrupt source is software-steerable to processor 2
- 32 software-settable interrupt request sources
- 10-bit vector
  - Unique vector for each interrupt request source
  - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 64 priorities
- Each interrupt source can be triggered by software
- Preemption
  - Preemptive prioritized interrupt requests to processor
  - ISR with higher priority preempts ISRs or tasks with lower priorities
  - Automatic pushing or popping of preempted priority to or from a LIFO
  - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
  - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
  - 3 INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; four clock cycles from receipt of software request

22.4 Modes of operation

The interrupt controller operates with the following handshaking modes: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN2], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ). In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

22.4.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for the processor when the HVEN2 bit in INTC_BCR (refer to *Section 22.5.2.1*) is negated. The hardware vector enable signal to the processor is driven as negated when the HVEN2 bit is negated. The vector is read from the INTC_IACKR2 register (refer to *Section 22.5.2.3*). Reading the INTC_IACKR2 negates the interrupt request to the processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in INTC_CPR2 (refer to *Section 22.5.2.2*) onto the LIFO, and updates PRI in INTC_CPR2 with the new priority.

22.4.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for the processor when the HVEN2 bit in INTC_BCR is asserted. The hardware vector enable signal to the processor is driven as asserted.

When the interrupt request to the processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKR2.

The assertion of the interrupt acknowledge signal for the processor pushes the PRI value in the INTC_CPR2 register onto the LIFO and updates the PRI in the INTC_CPR2 register with the new priority. This pushing of the PRI value onto the LIFO and updating PRI in INTC_CPR2 does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs (refer to *Section 22.5.2.5: INTC Software Set/Clear Interrupt Register n (INTC_SSCIRn)*) is written at a time such that the PRI value in the INTC_CPR2 register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in INTC_CPR2 is updated with the new priority, and the LIFO is neither pushed nor popped.

22.5 Memory map and register definition

22.5.1 Memory map

Table 246 is the INTC memory map.

The actual availability and number of registers are chip-specific. For this information, refer to the chapter that describes how modules are configured and connected.

Table 246. INTC memory map

Address offset	Register	Location
0x0000	INTC Block Configuration Register (INTC_BCR)	*Section 22.5.2.1*
0x0004–0x0007	Reserved(1)	
0x0008–0x000F	Reserved	
0x0010-0x0013	Reserved	
0x0014-0x0017	Reserved	
0x0018	INTC Current Priority Register for Processor 2 (INTC_CPR2)	*Section 22.5.2.2*
0x001C–0x001F	Reserved	
0x0020-0x0023	Reserved	
0x0024-0x0027	Reserved	
0x0028	INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)	*Section 22.5.2.3*
0x002C–0x002F	Reserved	

Table 246. INTC memory map (continued)

Address offset	Register	Location
0x0030-0x0033	Reserved	
0x0034-0x0037	Reserved	
0x0038	INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)	*Section 22.5.2.4*
0x003C–0x003F	Reserved	
0x0040–0x005F	INTC Software Set/Clear Interrupt Register 0–31 (INTC_SSCIR0–INTC_SSCIR31)	*Section 22.5.2.5*
0x0060–0x085F	INTC Priority Select Registers 0–1023 (INTC_PSR0–INTC_PSR1023)	*Section 22.5.2.6*
0x0860–0x0FFF	Reserved	

1. Access to this location will not result in transfer error.

Note: A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

22.5.2 Register descriptions

With the exception of the INTC_SSCIR*n* and INTC_PSR*n* registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8-bit wide, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSIR*n* are 8 bits wide and INTC_PSR*n* are 16-bit wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR2) are the same regardless of the size of the read. In software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR2) does not affect the operation of the write.

22.5.2.1 INTC Block Configuration Register (INTC_BCR)

The Block Configuration Register is used to configure options of the INTC.

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	Reserved	0	0	0	HVEN2	0	0	0	Reserved	0	0	0	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 165. INTC Block Configuration Register (INTC_BCR)

Table 247. INTC_BCR field descriptions

Field	Description
23 HVEN2	Hardware vector enable. Controls whether the INTC is in hardware vector mode or software vector mode. Refer to *Section 22.4*, for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

22.5.2.2 INTC Current Priority Register for Processor 2 (INTC_CPR2)

The INTC_CPR2 masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR2[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR2) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR2) is written, the LIFO is popped into the INTC_CPR2 PRI field. An exception case in hardware vector mode to this behavior is described in *Section 22.4.2*.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. Refer to *Section 22.7.5*.

Note: A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. Refer to Section 22.7.5.2, for example code to ensure coherency.

Offset: 0x0018 (INTC_CPR2) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PRI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 166. INTC Current Priority Register for Processor 2 (INTC_CPR2)

Table 248. INTC_CPR2 field descriptions

Field	Description
26:31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

22.5.2.3 INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)

The Interrupt Acknowledge Register for Processor 2 provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR2 has side effects from reads. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR2 does not have side effects in hardware vector mode.

Offset: 0x0028 (INTC_IACKR2) Access: User write always; non-speculative read(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	VTBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VTBA				INTVEC										0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. When INTC_BCR[HVEN2] = 1, a read of the INTC_IACKR2 has no side effects.

Figure 167. INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)

Table 249. INTC_IACKR2 field descriptions

Field	Description
0:19 VTBA	Vector table base address Must be set with the base address of the vector table prior to enabling interrupts.
20:29 INTVEC	Interrupt vector Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.

22.5.2.4 INTC End of Interrupt Register for Processor 2 (INTC_EOIR2)

Writing to the INTC_EOIR2 signals the end of the servicing of the interrupt request. When the INTC_EOIR2 is written, the priority last pushed on the LIFO is popped into INTC_CPR2. The values and size of data written to the INTC_EOIR2 are ignored. Those values and sizes written to this register neither update the INTC_EOIR2 contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

Offset: 0x0038 (INTC_EOIR2) Access: User write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	EOI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	EOI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 168. INTC End of Interrupt Register for Processor 2 (INTC_EOIR2)

Table 250. INTC_EOIR2 field descriptions

Field	Description
0:31 EOI	End of Interrupt Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

22.5.2.5 INTC Software Set/Clear Interrupt Register *n* (INTC_SSCIR*n*)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET will leave SET unchanged at 0 but will set CLR. Writing a 0 to SET will have no effect. CLR is the flag bit. Writing a 1 to CLR will clear it. Writing a 0 to CLR will have no effect. If a 1 is written to a pair of SET and CLR bits at the same time, CLR is asserted, regardless of whether CLR was asserted before the write.

Figure 169 shows the structure of the first INTC_SSCIR. The other registers follow the same structure.

Offset: 0x0040 + *n**0x1 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	CLR
W							SET	w1c
Reset	0	0	0	0	0	0	0	0

Figure 169. INTC Software Set/Clear Interrupt Register *n* (INTC_SSCIR*n*)

Table 251. INTC_SSCIR*n* field descriptions

Field	Description
6 SET	Set flag bit Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
7 CLR	Clear flag bit CLR is the flag bit. Writing a 1 to CLR will clear it provided that a 1 is not written simultaneously to its corresponding SET bit. Writing a 0 to CLR will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

22.5.2.6 INTC Priority Select Register *n* (INTC_PSR*n*)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSR*n* is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32–1023, and their priorities are configured in INTC_PSR32–1023, respectively.

Figure 170 and *Figure 171* show the structure and numbering scheme of the first and last PSRs, respectively. The other registers follow the same numbering scheme.

Note: Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.

Interrupts are not detected while the PSRs are being written.

Offset: 0x0060 + *n* * 0x2 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	Reserved	PRC_SEL2	Reserved	0	0	0	0	0	0	PRI					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 170. INTC Priority Select Register *n* (INTC_PSR*n*)

Table 252. INTC_PSR*n* field descriptions

Field	Description
2 PRC_SEL2	Processor 2 select bit If an interrupt source is enabled, PRC_SEL2 selects whether the interrupt request is to be sent to processor 2.
10:15 PRI	Priority select Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

Offset: 0x0060 + *n* * 0x2 (*n*=32 to 1023) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	Reserved	PRC_SEL2	Reserved	0	0	0	SWT	0	0	PRI					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 171. INTC Priority Select Register *n* (INTC_PSR*n*)

Table 253. INTC_PSR*n* field descriptions

Field	Description
2 PRC_SEL2	Processor 2 select bit If an interrupt source is enabled, PRC_SEL2 selects whether the interrupt request is to be sent to processor 2.
7 SWT	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a ‘1’ to SWT*n* sets the bit. On the interrupt acknowledge cycle, the SWT*n* bit is cleared.
10:15 PRI	Priority select Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

22.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

22.6.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI value in INTC_PSR*n* is higher than the PRI value in INTC_CPR2, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in INTC_CPR2 is updated with the corresponding PRI value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

22.6.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

22.6.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR*n*) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET bit in INTC_SSCIR*n*. This write sets the corresponding CLR bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR bit. Specific behavior includes the following:

- Writing a 1 to SET leaves SET unchanged at 0 but sets the flag bit (which is the CLR bit).
- Writing a 0 to SET has no effect.
- Writing a 1 to CLR clears the flag (CLR) bit.
- Writing a 0 to CLR has no effect.
- If a 1 is written to a pair of SET and CLR bits at the same time, the flag (CLR) is set, regardless of whether CLR was asserted before the write.

The time from the write to the SET bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

The processor will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the processor which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSR*n*[PRC_SEL2] bit.

22.6.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–31 are assigned vectors 0–31, respectively. The peripheral interrupt requests are assigned vectors from 32 to a number as high as needed to cover all peripheral interrupt requests.

22.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL2 values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in INTC_CPR2. The results of those comparisons are used to manage the priority of the ISR being executed by the processor. The LIFO also assists in managing that priority.

22.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in *Figure 164* are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for the processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for INTC_IACKR2, and if in hardware vector mode, for the interrupt vector provided to the processor.

22.6.2.1.1 Priority tree

The priority tree for the processor compares all the priorities of all of the asserted interrupt requests assigned to the processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to the processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the selector logic.

22.6.2.1.2 Selector

If only one interrupt request from the priority tree is asserted, then it is passed as asserted to the encoder logic. If multiple interrupt requests from the priority tree are asserted, then only the one with the lowest vector is passed as asserted to the encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

22.6.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the processor.

22.6.2.1.4 Comparator

The comparator logic compares the highest priority output from the priority arbitrator subblock with PRI in INTC_CPR2. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in INTC_CPR2, or the PRI value in INTC_CPR2 is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in INTC_CPR2 when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI in INTC_PSR*n* are zero will not cause a preemption because their PRI will not be higher than PRI in INTC_CPR2.

Another function of the comparator is to signal an update of INTC_IACKR2 with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of INTC_CPR2 occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

22.6.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from INTC_CPR2. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in INTC_CPR2 does not need to be loaded from INTC_CPR2 and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored in INTC_CPR2.

The PRI value in INTC_CPR2 is pushed onto the LIFO when INTC_IACKR2 is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode. The priority is popped into PRI in INTC_CPR2 whenever INTC_EOIR2 is written. An exception case in hardware vector mode to this behavior is described in *Section 22.4.2*.

Although the INTC supports up to 64 priorities, an ISR executing with PRI in INTC_CPR2 equal to 63 will not be preempted. Therefore, the LIFO supports the stacking of 63 priorities. However, the LIFO is only 62 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 63 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

22.6.3 Handshaking with processor

22.6.3.1 Software vector mode handshaking

22.6.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to *Figure 172*.

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in *Figure 172*. The INTC examines the peripheral and software-settable interrupt requests.

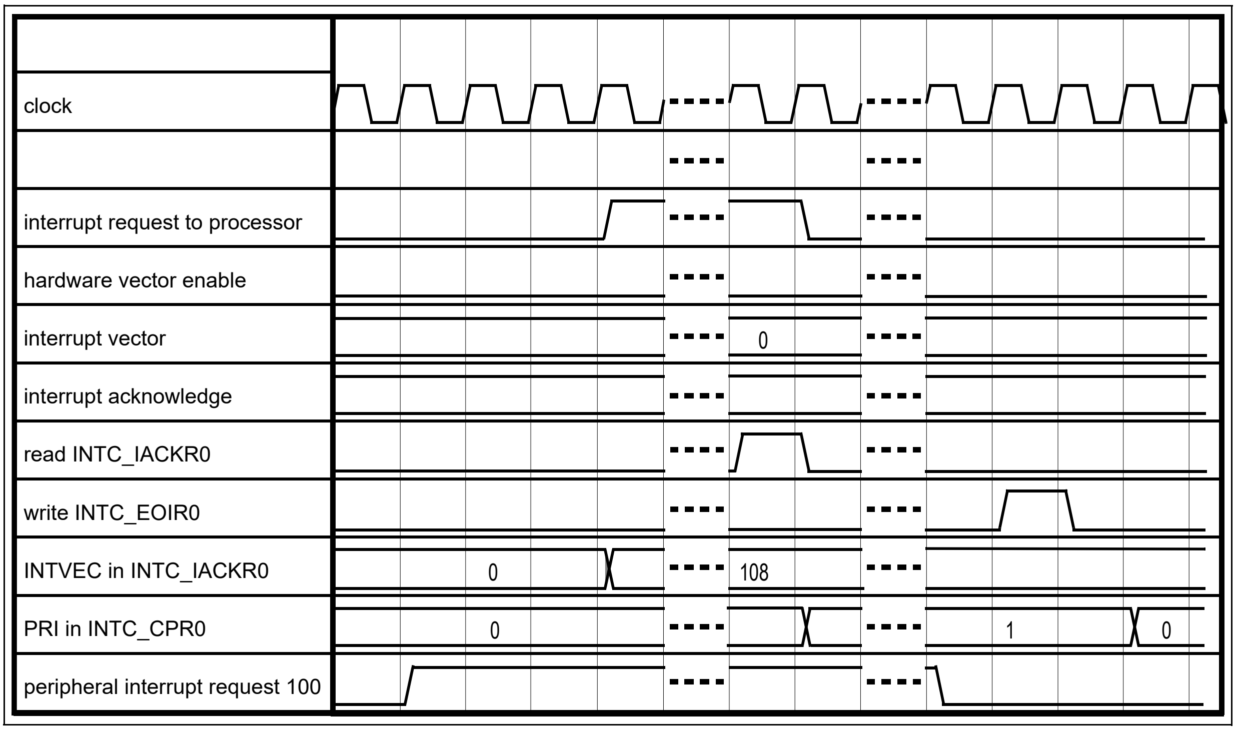
When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in INTC_CPR2, it asserts the interrupt request to the processor. The INTVEC field in INTC_IACKR2 is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in *Section 22.4.1*.

22.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR2 must be written. When it is written, the LIFO is popped so that the preempted priority is restored into PRI of INTC_CPR2. Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in INTC_CPR2 is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

Figure 172. Timing diagram of software vector mode handshaking

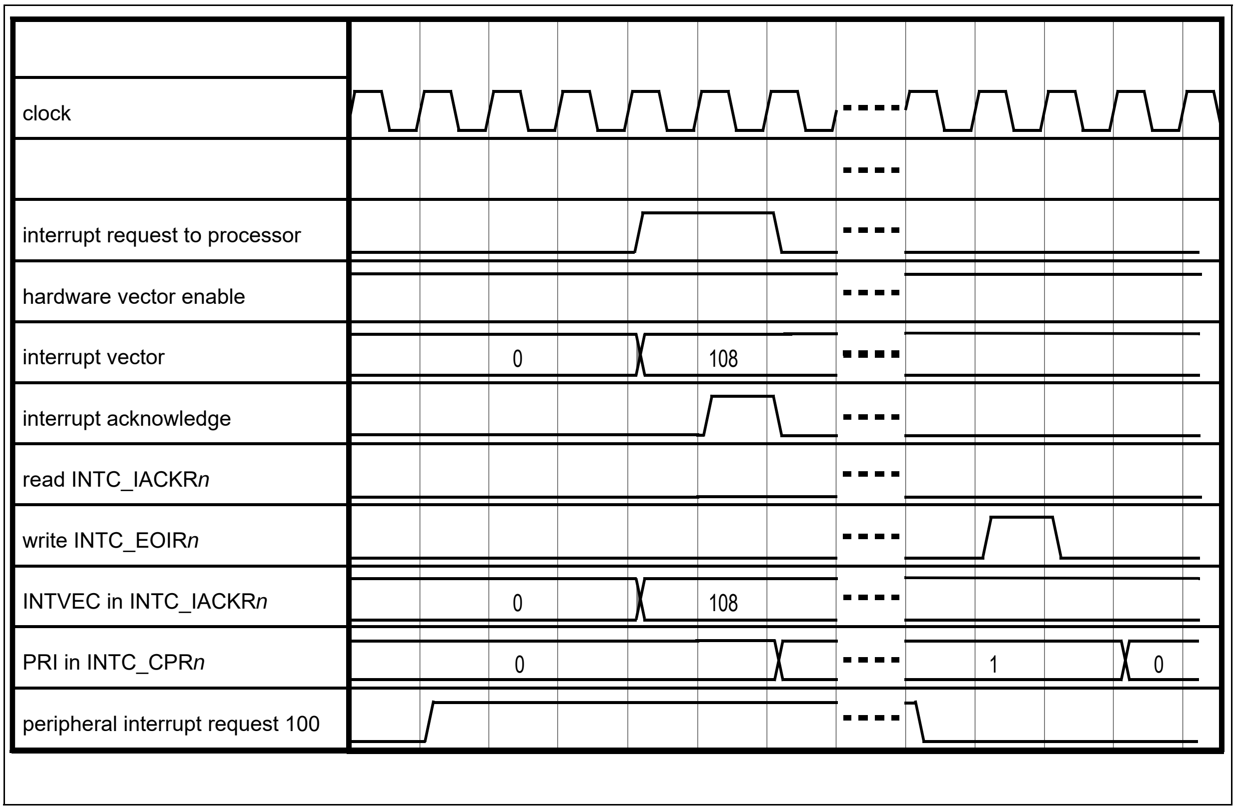


22.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in *Figure 173*. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in INTC_CPR2, it asserts the interrupt request to the processor. The INTVEC field in INTC_IACKR2 is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. In addition, the value of the interrupt vector to the processor matches the value of the INTVEC field in INTC_IACKR2. The rest of the handshaking is described in *Section 22.4.2*.

The handshaking near the end of the interrupt exception handler, that is the writing to INTC_EOIR2, is the same as in software vector mode. Refer to *Section 22.6.3.1.2: End of interrupt exception handler*.

Figure 173. Timing diagram for hardware vector mode handshaking



22.7 Initialization/application information

22.7.1 Initialization flow

After exiting reset, all of the PRI and PRC_SEL2 fields in INTC_PSR*n* are zero, and PRI in the INTC_CPR2 register is 31. These reset values will prevent the INTC from asserting the interrupt request to the processor. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- Interrupt_request_initialization:
  - Configure HVEN2 in INTC_BCR.
  - Configure VTBA in INTC_IACKR2.
  - Raise the PRI fields and set the PRC_SEL2 fields in INTC_PSR*n*.
  - Set the enable bits or clear the mask bits for the peripheral interrupt requests.
  - Lower PRI in INTC_CPR2 to zero.
  - Enable processor recognition of interrupts.

22.7.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

22.7.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 bytes of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
b  interrupt_exception_handler_continued
   # 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1
(Power Architecture Save/Restore registers) (not shown)

lis     r3,INTC_IACKR2@ha    # form adjusted upper half of INTC_IACKR2
                             # address
lwz     r3,INTC_IACKR2@l(r3) # load INTC_IACKR2, which clears request to
                             # processor
lwz     r3,0x0(r3)           # load address of ISR from vector table
wrteei  1                    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded
Binary Application Interface) (not shown)

mtlr    r3           # move the INTC_IACKR2 address into the link register
blrl                 # branch to ISR; link register updated with epilog
                     # address

epilog:
code to restore most of context required by Power Architecture EABI (not
shown)
```

```
# Popping the LIFO after the restoration of most of the context and the
# disabling of processor recognition of interrupts eases the calculation of
# the maximum stack depth at the cost of postponing the servicing of the
# next interrupt request.
mbar                          # ensure store to clear flag bit has completed
lis     r3,INTC_EOIR2@ha      # form adjusted upper half of INTC_EOIR2
                              # address
li      r4,0x0                # form 0 to write to INTC_EOIR2
wrteei  0                     # disable processor recognition of interrupts
stw     r4,INTC_EOIR2@l(r3)   # store to INTC_EOIR2, informing INTC to lower
                              # priority

code to restore SRR0 and SRR1, restore working registers, and delete stack
frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
   .
   .
   .
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023

ISRn:
code to service the interrupt event (not shown)
code to clear flag bit which drives interrupt request to INTC (not shown)

blr        # return to epilog
```

22.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b  interrupt_exception_handler_continuedn  # 16 bytes available, branch to
                                           # continue

interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1
(not shown)

wrteei  1       # enable processor recognition of interrupts
```

```
code to save rest of context required by Power Architecture EABI (not shown)

bl      ISRn    # branch to ISR for interrupt with vector n

epilog:
code to restore most of context required by Power Architecture EABI (not
shown)

# Popping the LIFO after the restoration of most of the context and the
# disabling of processor recognition of interrupts eases the calculation of
# the maximum stack depth at the cost of postponing the servicing of the
# next interrupt request.
mbar                         # ensure store to clear flag bit has completed
lis     r3,INTC_EOIR2@ha     # form adjusted upper half of INTC_EOIR2
                             # address
li      r4,0x0               # form 0 to write to INTC_EOIR2
wrteei  0                    # disable processor recognition of interrupts
stw     r4,INTC_EOIR2@l(r3)  # store to INTC_EOIRn, informing INTC to lower
priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr# branch to epilog
```

22.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR2 having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR2 priority 0 and outside the control of the RTOS while RTOS executes at INTC_CPR2 priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPR2 priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task’s priority can be elevated in the INTC_CPR2 while the shared resource is being accessed.

An ISR whose PRI in INTC_PSR*n* has a value of 0 will not cause an interrupt request to the processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

22.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in *Table 254* shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 254. Order of ISR execution example

		Code executing at end of step						
Step#	**Step Description**	**RTOS**	**ISR108[1]**	**ISR208**	**ISR308**	**ISR408**	**interrupt exception handler**	**PRI in INTC_CPR at end of step**
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR2.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR2.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3

Table 254. Order of ISR execution example (continued)

		Code executing at end of step						
Step#	Step Description	RTOS	ISR108(1)	ISR208	ISR308	ISR408	interrupt exception handler	PRI in INTC_CPR at end of step
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR2.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR2.						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

22.7.5 Priority ceiling protocol

22.7.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR2) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR2, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR2 to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

22.7.5.2 Ensuring coherency

22.7.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR2. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR2 can be made by those system services with the code sequence:

Figure 174. OSEK Code Sequence

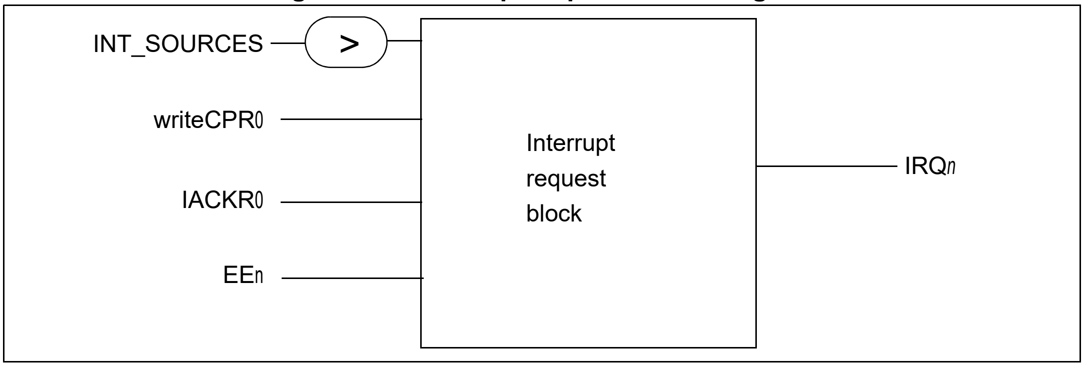
```
GetResource:
   wrteei  0        # disable external interrupts to the Processor
   raise   PRI      # Write to CPR, cache inhibit, guarded
   mbar             # flush out writes from store buffer
   wrteei  1        # enable external interrupts to the Processor
   isync            # re-fetch Processor pipeline

ReleaseResource:
   mbar             # flush out writes from store buffer
   wrteei  0        # disable external interrupts to the Processor
   lower   PRI      # Write to CPR, cache inhibit, guarded
   wrteei  1        # enable external interrupts to the Processor
```

22.7.5.2.1.1 Interrupt request to processor

Referencing *Figure 164*, the interrupt request logic to the processor is shown in *Figure 175*.

Figure 175. Interrupt request block diagram



Assuming that there are no IRQ*n* deasserted (no pending interrupt requests to the processor), if one of the external interrupt sources (INT_SOURCES) has a priority higher than INTC_CPR2, then interrupt request to the processor (IRQ*n*) is asserted. It will stay asserted until one of the following conditions is true:

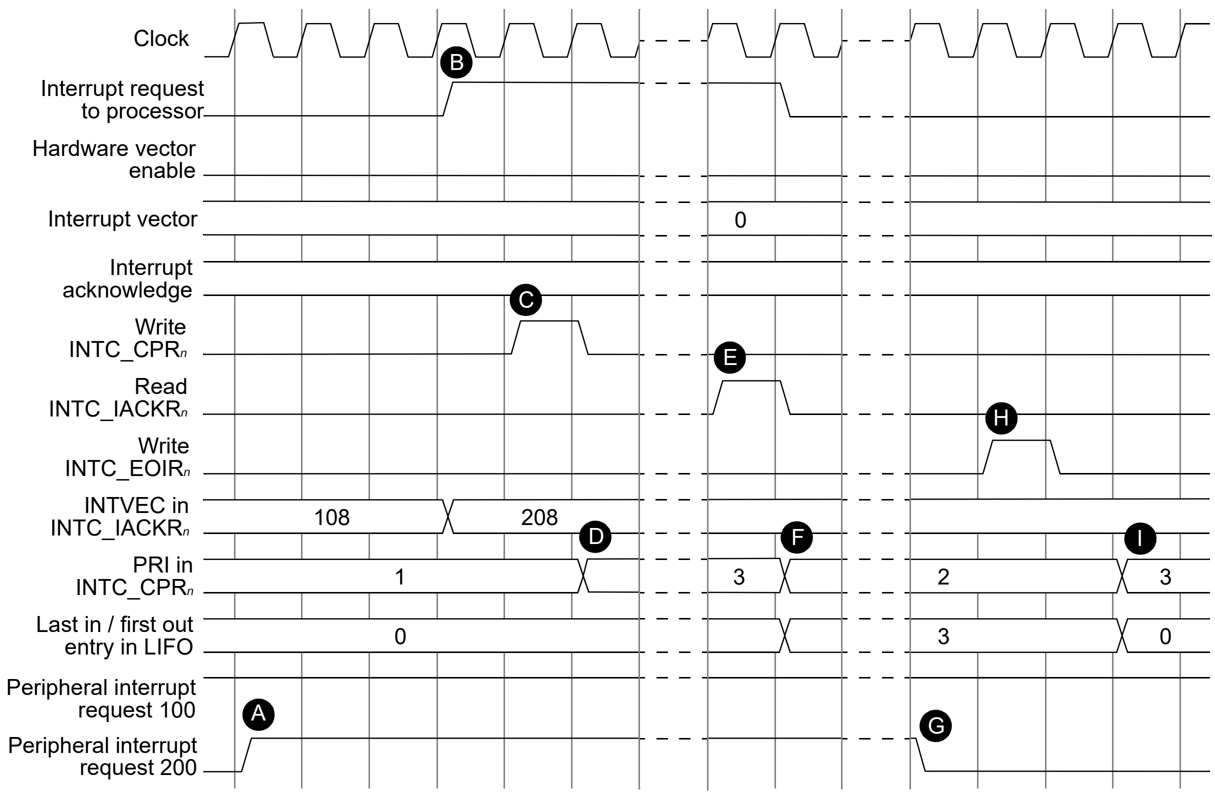
- Interrupt acknowledge (IACKR2) is asserted
- If the EE*n* bit has been cleared by the wrteei instruction (refer to *Figure 174*), IRQ*n* will be re-evaluated when the processor writes to INTC_CPR2 while the processor ignores IRQ*n*

This provides a safe way to guarantee that no interrupts will be recognized by the processor while updating INTC_CPR2.

22.7.5.2.2 Raised priority preserved

Before the instruction and after the GetResource system service executes, all pending transactions are complete. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. Also, during the epilog of the interrupt exception handler for this preempting ISR, the raised priority has been restored from the LIFO to PRI in INTC_CPR2. The shared coherent data block now can be accessed coherently. *Figure 176* shows the timing diagram for this scenario, and *Table 255* explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

Figure 176. Timing diagram of raised priority preserved



Note that n = 2 for above registers

Table 255. Raised priority preserved events

Event	Description
A	Peripheral interrupt request with name equal to 200, asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKR2 updates with vector for that peripheral interrupt request.
C	ISR108 writes to INTC_CPR2 to raise priority to 3 before accessing shared coherent data block.

Table 255. Raised priority preserved events (continued)

Event	Description
D	PRI in INTC_CPR2 now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
E	Interrupt exception handler prolog acknowledges interrupt by reading INTC_IACKR2.
F	PRI of 3 pushed onto LIFO. PRI in INTC_CPR2 updates to 2, the priority of ISR208.
G	ISR208 clears its flag bit, deasserting its peripheral interrupt request.
H	Interrupt exception handler epilog writes to INTC_EOIR2.
I	LIFO pops 3, restoring the raised priority onto PRI in INTC_CPR2. Next value to pop from LIFO is the priority from before peripheral interrupt request with name equal to 100 interrupted. ISR108 now can access data block coherently after interrupt exception handler executes rfi instruction.

22.7.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 64 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor’s ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

22.7.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR.

22.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRI value in INTC_PSR*n*, which becomes the PRI value in INTC_CPR2 with the interrupt acknowledge. The ISR, however, can have a portion of it which need not be executed at this higher priority. Therefore, executing this later portion which need not be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET bit in INTC_SSCIR*n*. Writing a 1 to SET causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

22.7.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR*n*, one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in *Section 22.7.7.1*) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note: Lowering the PRI value in INTC_CPR2 within an ISR to below the ISR's corresponding PRI value in INTC_PSRn allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

22.7.9 Negating an interrupt request outside of its ISR

22.7.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

22.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

22.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI values in INTC_PSR*n* must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR*n* as the clearing of the flag bit that caused the present ISR to be executed. Refer to *Section 22.6.3.1.2* for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI value in INTC_PSR*n*.

22.7.10 Examining LIFO contents

Normally the user does not need to know the contents of the LIFO, or even how deep the LIFO is nested. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR2). Disabling processor recognition of interrupts while examining the LIFO contents provides a coherent view of the preempted priorities. The code sequence is:

```
pop_lifo:
   store to INTC_EOIR2
   load INTC_CPR2, examine PRI, and store onto stack
   if PRI is not zero or value when interrupts were enabled, branch to
pop_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
   load stacked PRI value and store to INTC_CPR2
   load INTC_IACKR2
   if stacked PRI values are not depleted, branch to push_lifo
```

Note: Reading the INTC_IACKR2 acknowledges the interrupt request to the processor and updates the INTC_CPR2[PRI] with the priority of the preempting interrupt request. If the processor recognition of interrupts is disabled during the LIFO restoration, interrupt requests to the processor can go undetected. However, because the peripheral or software-settable interrupt requests are not cleared, the peripheral interrupt request to the processor re-asserts when INTC_CPR2[PRI] is lower than the priorities of those peripheral or software-settable interrupt requests.

22.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, refer to the chapter that describes how modules are configured and connected.

23 Enhanced Direct Memory Access (eDMA)

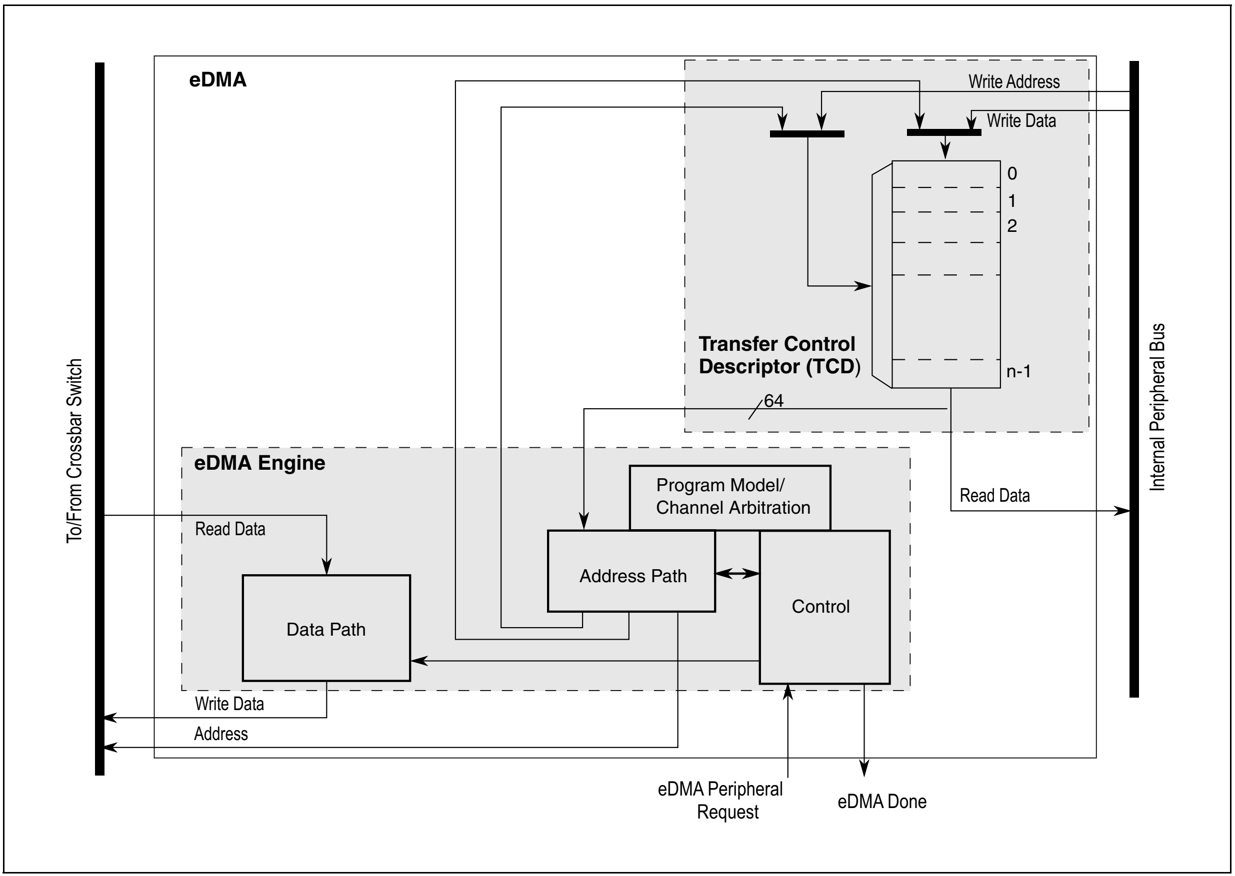
Note: For the chip-specific implementation details of this module's instances, please refer to the device configuration section.

23.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes a DMA engine that performs source- and destination-address calculations, and the actual data-movement operations, along with local memory containing transfer control descriptors for each of the 32 channels.

Figure 177 illustrates the eDMA module.

Figure 177. eDMA block diagram



23.1.1 Features

The eDMA is a highly-programmable data-transfer engine optimized to minimize the intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the data packet itself. The eDMA module features:

- Data movements via dual-address transfers: read from source, write to destination:
  - Programmable source and destination addresses and transfer size, plus support for enhanced addressing modes.
- 32-channel implementation that performs complex data transfers with minimal intervention from a host processor.
  - Internal data buffer, used as temporary storage to support 16- and 32-byte burst transfers.
  - Connections to the crossbar switch for bus mastering the data movements.
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations.
  - 32-byte TCD stored in local memory for each channel.
  - An inner data transfer loop defined by a minor byte transfer count.
  - An outer data transfer loop defined by a major iteration count.
- Channel activation via one of three methods:
  - Explicit software initiation.
  - Initiation via a channel-to-channel linking mechanism for continuous transfers.
  - Peripheral-paced hardware requests (one per channel).
- Fixed-priority and round-robin channel arbitration.
- Channel completion reported via optional interrupt requests.
  - One interrupt per channel, optionally asserted at completion of major iteration count.
  - Optional error terminations per channel and logically summed together to form one error interrupt to the interrupt controller.
- Optional support for scatter/gather DMA processing.
- Support for complex data structures.
- Support to cancel transfers via software.

Throughout this chapter, *n* is used to reference the channel number.

23.2 Modes of operation

The eDMA supports the following modes of operation.

23.2.1 Normal mode

In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.

A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.

23.2.2 Debug mode

DMA operation is configurable in Debug mode via the control register:

- If CR[EDBG] is cleared, the DMA continues to operate.
- If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.

23.2.3 Wait mode

Before entering Wait mode the DMA attempts to complete its current transfer. After the transfer completes the device enters Wait mode.

23.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions.
- The second region corresponds to the local transfer control descriptor memory.

Some registers are implemented as two 32-bit registers, and include an H and L suffix, signaling the high and low portions of the control function. *Table 256* provides a 32-bit view of the eDMA's memory map.

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0 to channel 31. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

Reading reserved bits in a register returns the value of zero and writes to reserved bits in a register are ignored. Reading or writing a reserved memory location generates a bus error.

Table 256. DMA memory map

Address offset	Register name	Location
0x0000	Control Register (eDMA_CR)	*Section 23.3.1*
0x0004	Error Status Register (eDMA_ES)	*Section 23.3.2*
0x0008–0x000B	Reserved	
0x000C	Enable Request Register Low (eDMA_ERQL)	*Section 23.3.3*
0x0010–0x0013	Reserved	
0x0014	Enable Error Interrupt Register Low (eDMA_EEIL)	*Section 23.3.4*
0x0018	Set Enable Request Register (eDMA_SERQ)	*Section 23.3.5*
0x0019	Clear Enable Request Register (eDMA_CERQ)	*Section 23.3.6*
0x001A	Set Enable Error Interrupt Register (eDMA_SEEI)	*Section 23.3.7*
0x001B	Clear Enable Error Interrupt Register (eDMA_CEEI)	*Section 23.3.8*
0x001C	Clear Interrupt Request Register (eDMA_CINT)	*Section 23.3.9*
0x001D	Clear Error Register (eDMA_CERR)	*Section 23.3.10*
0x001E	Set START Bit Register (eDMA_SSRT)	*Section 23.3.11*

Table 256. DMA memory map (continued)

Address offset	Register name	Location
0x001F	Clear DONE Status Bit Register (eDMA_CDNE)	Section 23.3.12
0x0020–0x0023	Reserved	
0x0024	Interrupt Request Register Low (eDMA_INTL)	Section 23.3.13
0x0028–0x002B	Reserved	
0x002C	Error Register Low (eDMA_ERRL)	Section 23.3.14
0x0030–0x0033	Reserved	
0x0034	Hardware Request Status Register Low (eDMA_HRSL)	Section 23.3.15
0x0038–0x00FF	Reserved	
eDMA channel registers per channel (*n* = 0 to 31)		
0x0100 + *n**0x1	Channel *n* Priority Register (eDMA_DCHPRI*n*)	Section 23.3.16
0x0120–0x013F	Reserved	
0x0140 + *n**0x1	Channel *n* Master ID Register (eDMA_DCHMID*n*)	Section 23.3.17
0x0160–0x0FFF	Reserved	
0x1000+*n**0x20	TCD Source Address (eDMA_TCD*n*_SADDR)	Section 23.3.18
0x1004+*n**0x20	TCD Transfer Attributes (eDMA_TCD*n*_ATTR)	Section 23.3.19
0x1006+*n**0x20	TCD Signed Source Address Offset (eDMA_TCD*n*_SOFF)	Section 23.3.20
0x1008+*n**0x20	TCD Minor Byte Count (Minor Loop Disabled) (eDMA_TCD*n*_NBYTES_MLNO)	Section 23.3.21
	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (eDMA_TCD*n*_NBYTES_MLOFFNO)	Section 23.3.22
	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (eDMA_TCD*n*_NBYTES_MLOFFYES)	Section 23.3.23
0x100C+*n**0x20	TCD Last Source Address Adjustment (eDMA_TCD*n*_SLAST)	Section 23.3.24
0x1010+*n**0x20	TCD Destination Address (eDMA_TCD*n*_DADDR)	Section 23.3.25
0x1014+*n**0x20	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_CITER_ELINKYES)	Section 23.3.26
	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_CITER_ELINKNO)	Section 23.3.27
0x1016+*n**0x20	TCD Signed Destination Address Offset (eDMA_TCD*n*_DOFF)	Section 23.3.28
0x1018+*n**0x20	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD*n*_DLASTSGA)	Section 23.3.29
0x101C+*n**0x20	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_BITER_ELINKYES)	Section 23.3.30
	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_BITER_ELINKNO)	Section 23.3.31
0x101E+*n**0x20	TCD Control and Status (eDMA_TCD*n*_CSR)	Section 23.3.32
0x1400–0x17FF	Reserved	

23.3.1 Control Register (eDMA_CR)

The CR defines the basic operating configuration of the DMA.

The DMA arbitrates channel service requests in four groups (0, 1, 2, 3) of 16 channels each. Group 3 contains channels 63–48; group 2 contains channels 47–32; group 1 contains channels 31–16; and group 0 contains channels 15–0.

Arbitration within a group can be configured to use a fixed-priority or a round-robin scheme. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities (refer to the DCHPRI*n* registers). In Round-Robin Arbitration mode, the channel priorities are ignored, and channels within each group are cycled through without regard to priority.

Note: Writes to the DMA_CR must be performed only when the DMA channels are inactive (TCDn_CSR[ACTIVE] bits are cleared).

The group priorities operate in a similar fashion. In Group Fixed-Priority Arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRP*n*PRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to the occurrence of any channel service requests; otherwise, a configuration error will be reported. In Group Round-Robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Address: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														CX	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	GRP1PRI		GRP0PRI		EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	0
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 178. Control Register (eDMA_CR)

Table 257. eDMA_CR field descriptions

Field	Description
14 CX	Cancel transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error cancel transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to canceling the transfer, ECX treats the cancel as an error condition; thus updating the ES register and generating an optional error interrupt.
20:21 GRP1PRI	Channel group 1 priority Group 1 priority level when fixed-priority group arbitration is enabled.
22:23 GRP0PRI	Channel group 0 priority Group 0 priority level when fixed-priority group arbitration is enabled.
24 EMLM	Enable minor loop mapping 0 Disabled. TCD*n*.word2 is defined as a 32-bit NBYTES field. 1 Enabled. TCD*n*.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.
25 CLM	Continuous Link mode 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt on error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable round-robin group arbitration 0 Fixed-priority arbitration is used for selection among the groups. 1 Round-robin arbitration is used for selection among the groups.

Table 257. eDMA_CR field descriptions (continued)

Field	Description
29 ERCA	Enable round-robin channel arbitration 0 Fixed-priority arbitration is used for channel selection within each group. 1 Round-robin arbitration is used for channel selection within each group.
30 EDBG	Enable debug 0 When in Debug mode the DMA continues to operate. 1 When in Debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system or the EDBG bit is cleared.

23.3.2 Error Status Register (eDMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode) or an error termination to a bus master read or write cycle.

Refer to *Section 23.4.3*, for more details.

Address: 0x0004 Access: User read-write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 179. Error Status Register (eDMA_ES)

Table 258. eDMA_ES field descriptions

Field	Description
0 VLD	ERRH and ERRL status bits 0 No ERR bits are set. 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
14 UCE	Uncorrectable ECC error 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error.
15 ECX	Transfer canceled 0 No canceled transfers. 1 The last recorded entry was a zeroed transfer by the error cancel transfer input.
16 GPE	Group priority error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.

Table 258. eDMA_ES field descriptions (continued)

Field	Description
17 CPE	Channel priority error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities. Channel priorities are not unique.
18:23 ERRCHN	Error channel number or canceled channel number GPE or last recorded error canceled transfer.
24 SAE	Source address error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCD*n*_SADDR field. TCD*n*_SADDR is inconsistent with TCD*n*_ATTR[SSIZE].
25 SOE	Source offset error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_SOFF field. TCD*n*_SOFF is inconsistent with TCD*n*_ATTR[SSIZE].
26 DAE	Destination address error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DADDR field. TCD*n*_DADDR is inconsistent with TCD*n*_ATTR[DSIZE].
27 DOE	Destination offset error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DOFF field. TCD*n*_DOFF is inconsistent with TCD*n*_ATTR[DSIZE].
28 NCE	NBYTES/CITER configuration error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_NBYTES or TCD*n*_CITER fields.
29 SGE	Scatter/gather configuration error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD*n*_CSR[ESG] is enabled. TCD*n*_DLASTSGA is not on a 32-byte boundary.
30 SBE	Source bus error 0 No source bus error 1 The last recorded error was a bus error on a source read.
31 DBE	Destination bus error 0 No destination bus error 1 The last recorded error was a bus error on a destination write.

23.3.3 Enable Request Register Low (eDMA_ERQL)

The ERQL register provides a bit map for the 32 implemented channels to enable the request signal for each channel.

The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQL.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ 31	ERQ 30	ERQ 29	ERQ 28	ERQ 27	ERQ 26	ERQ 25	ERQ 24	ERQ 23	ERQ 22	ERQ 21	ERQ 20	ERQ 19	ERQ 18	ERQ 17	ERQ 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ 15	ERQ 14	ERQ 13	ERQ 12	ERQ 11	ERQ 10	ERQ 9	ERQ 8	ERQ 7	ERQ 6	ERQ 5	ERQ 4	ERQ 3	ERQ 2	ERQ 1	ERQ 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 180. Enable Request Register Low (eDMA_ERQL)

Table 259. eDMA_ERQL field descriptions

Field	Description
0:31 ERQ[31:0]	Enable DMA Request x 0 The DMA request signal for the corresponding channel is disabled. 1 The DMA request signal for the corresponding channel is enabled.

23.3.4 Enable Error Interrupt Register Low (eDMA_EEIL)

The Enable Error Interrupt Register Low (DMA_EEIL) provides a bit map for the 32 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEIL register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI 31	EEI 30	EEI 29	EEI 28	EEI 27	EEI 26	EEI 25	EEI 24	EEI 23	EEI 22	EEI 21	EEI 20	EEI 19	EEI 18	EEI 17	EEI 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI 15	EEI 14	EEI 13	EEI 12	EEI 11	EEI 10	EEI 9	EEI 8	EEI 7	EEI 6	EEI 5	EEI 4	EEI 3	EEI 2	EEI 1	EEI 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 181. Enable Error Interrupt Register Low (eDMA_EEIL)

Table 260. eDMA_EEIL field descriptions

Field	Description
0:31 EEI[31:0]	Enable error interrupt x 0 The error signal for corresponding channel does not generate an error interrupt. 1 The assertion of the error signal for corresponding channel generates an error interrupt request.

23.3.5 Set Enable Request Register (eDMA_SERQ)

The Set Enable Request Register (DMA_SERQ) provides a simple memory-mapped mechanism to set a given bit in the ERQL to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQL to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQL to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x0018 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAER	SERQ					
Reset	0	0	0	0	0	0	0	0

Figure 182. Set Enable Request Register (eDMA_SERQ)

Table 261. eDMA_SERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAER	Set all enable requests 0 Set only those ERQ bits specified in the SERQ field 1 Set all bits in ERQL
2:7 SERQ	Set enable request Sets the corresponding bit in ERQL

23.3.6 Clear Enable Request Register (eDMA_CERQ)

The Clear Enable Request Register (DMA_CERQ) provides a simple memory-mapped mechanism to clear a given bit in the ERQL to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQL to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQL to be cleared, disabling all DMA request inputs.

If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x0019 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAER	CERQ					
Reset	0	0	0	0	0	0	0	0

Figure 183. Clear Enable Request Register (eDMA_CERQ)

Table 262. eDMA_CERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAER	Clear all enable requests 0 Clear only those ERQ bits specified in the CERQ field 1 Clear all bits in ERQ
2:7 CERQ	Clear enable request Clears the corresponding bit in ERQL

23.3.7 Set Enable Error Interrupt Register (eDMA_SEEI)

The Set Enable Error Interrupt Register (DMA_SEEI) provides a simple memory-mapped mechanism to set a given bit in the EEIL to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEIL to be set. Setting the SAEE bit provides a global set function, forcing the entire EEIL contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001A Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAEE	SEEI					
Reset	0	0	0	0	0	0	0	0

Figure 184. Set Enable Error Interrupt Register (eDMA_SEEI)

Table 263. eDMA_SEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAEE	Sets all enable error interrupts 0 Set only those EEI bits specified in the SEEI field. 1 Sets all bits in EEI.
2:7 SEEI	Set enable error interrupt Sets the corresponding bit in EEIL

23.3.8 Clear Enable Error Interrupt Register (eDMA_CEEI)

The Clear Enable Error Interrupt Register (DMA_CEEI) provides a simple memory-mapped mechanism to clear a given bit in the EEIL to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEIL to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEIL contents to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001B Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAEE	CEEI					
Reset	0	0	0	0	0	0	0	0

Figure 185. Clear Enable Error Interrupt Register (eDMA_CEEI)

Table 264. eDMA_CEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAEE	Clear all enable error interrupts 0 Clear only those EEI bits specified in the CEEI field 1 Clear all bits in EEI
2:7 CEEI	Clear enable error interrupt Clears the corresponding bit in EEIL

23.3.9 Clear Interrupt Request Register (eDMA_CINT)

The Clear Interrupt Request Register (DMA_CINT) provides a simple, memory-mapped mechanism to clear a given bit in the INTL to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INTL to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INTL to be cleared, disabling all DMA interrupt requests.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001C Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAIR	CINT					
Reset	0	0	0	0	0	0	0	0

Figure 186. Clear Interrupt Request Register (eDMA_CINT)

Table 265. eDMA_CINT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAIR	Clear all interrupt requests 0 Clear only those INT bits specified in the CINT field 1 Clear all bits in INTL
2:7 CINT	Clear interrupt request Clears the corresponding bit in INTL

23.3.10 Clear Error Register (eDMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERRL to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERRL to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERRL contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001D Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAEI	CERR					
Reset	0	0	0	0	0	0	0	0

Figure 187. Clear Error Register (eDMA_CERR)

Table 266. eDMA_CERR field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAEI	Clear all error indicators 0 Clear only those ERR bits specified in the CERR field 1 Clear all bits in ERR
2:7 CERR	Clear error indicator Clears the corresponding bit in ERRL

23.3.11 Set START Bit Register (eDMA_SSRT)

The Set START Bit Register (DMA_SSRT) provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001E Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAST	SSRT					
Reset	0	0	0	0	0	0	0	0

Figure 188. Set START Bit Register (eDMA_SSRT)

Table 267. eDMA_SSRT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAST	Set all START bits (activates all channels) 0 Set only those TCD*n*_CSR[START] bits specified in the SSRT field 1 Set all bits in TCD*n*_CSR[START]
2:7 SSRT	Set START bit Sets the corresponding bit in TCD*n*_CSR[START]

23.3.12 Clear DONE Status Bit Register (eDMA_CDNE)

The Clear DONE Status Bit Register (DMA_CDNE) provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001F Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CADN	CDNE					
Reset	0	0	0	0	0	0	0	0

Figure 189. Clear DONE Status Bit Register (eDMA_CDNE)

Table 268. eDMA_CDNE field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CADN	Clears all DONE bits 0 Clears only those TCD*n*_CSR[DONE] bits specified in the CDNE field 1 Clears all bits in TCD*n*_CSR[DONE]
2:7 CDNE	Clear DONE bit Clears the corresponding bit in TCD*n*_CSR[DONE]

23.3.13 Interrupt Request Register Low (eDMA_INTL)

The Interrupt Request Register Low (DMA_INTL) provides a bit map for the 32 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt at data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT. On writes to the INT, a '1' in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The CINT is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INTL.

Address: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT 31	INT 30	INT 29	INT 28	INT 27	INT 26	INT 25	INT 24	INT 23	INT 22	INT 21	INT 20	INT 19	INT 18	INT 17	INT 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT 15	INT 14	INT 13	INT 12	INT 11	INT 10	INT 9	INT 8	INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 190. Interrupt Request Register Low (eDMA_INTL)

Table 269. eDMA_INTL field descriptions

Field	Description
0:31 INT[31:0]	Interrupt request x 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.

23.3.14 Error Register Low (eDMA_ERRL)

The ERRL provides a bit map for the 32 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16 and 32 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR 31	ERR 30	ERR 29	ERR 28	ERR 27	ERR 26	ERR 25	ERR 24	ERR 23	ERR 22	ERR 21	ERR 20	ERR 19	ERR 18	ERR 17	ERR 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR 15	ERR 14	ERR 13	ERR 12	ERR 11	ERR 10	ERR 9	ERR 8	ERR 7	ERR 6	ERR 5	ERR 4	ERR 3	ERR 2	ERR 1	ERR 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 191. Error Register Low (eDMA_ERRL)

Table 270. eDMA_ERRL field descriptions

Field	Description
0:31 ERR[31:0]	Error in channel x 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.

23.3.15 Hardware Request Status Register Low (eDMA_HRSL)

The HRSL provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. The Hardware Request Status bits reflect the current state of the register and qualified (via the ERQL fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

Note: These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQL bits.

Address: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	HRS 31	HRS 30	HRS 29	HRS 28	HRS 27	HRS 26	HRS 25	HRS 24	HRS 23	HRS 22	HRS 21	HRS 20	HRS 19	HRS 18	HRS 17	HRS 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	HRS 15	HRS 14	HRS 13	HRS 12	HRS 11	HRS 10	HRS 9	HRS 8	HRS 7	HRS 6	HRS 5	HRS 4	HRS 3	HRS 2	HRS 1	HRS 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 192. Hardware Request Status Register Low (eDMA_HRSL)

Table 271. eDMA_HRSL field descriptions

Field	Description
0:31 HRS[31:0]	Hardware request status channel x 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.

23.3.16 Channel *n* Priority Register (eDMA_DCHPRI*n*)

When the Fixed-Priority Channel Arbitration mode is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values. Otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15.

When read, the GRPPRI bits of the DCHPRI*n* register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRI*n* registers. The group priority is assigned in the DMA control register.

Address: 0x0100 + *n**0x1 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7
R	ECP	DPA	GRPPRI		CHPRI			
W								
Reset(1)	0	0	x	x	0	0	0	0

1. x = Undefined at reset.

Figure 193. Channel *n* Priority Register (eDMA_DCHPRI*n*)

Table 272. eDMA_DCHPRI*n* field descriptions

Field	Description
0 ECP	Enable channel preemption This bit resets to zero. 0 Channel *n* cannot be suspended by a higher priority channel's service request. 1 Channel *n* can be temporarily suspended by the service request of a higher priority channel.
1 DPA	Disable preempt ability This bit resets to zero. 0 Channel *n* can suspend a lower priority channel. 1 Channel *n* cannot suspend any channel, regardless of channel priority.
2:3 GRPPRI	Channel *n* current group priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. **Note:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.
4:7 CHPRI	Channel *n* arbitration priority Channel priority when fixed-priority arbitration is enabled. **Note:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.

23.3.17 Channel *n* Master ID Register (eDMA_DCHMID*n*)

The DMA Master ID Replication registers allow the DMA to use the same protection level and AHB system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master (a core, for example) programs a TCD, its master ID and protection level are captured when the TCD Word 7 control attributes are written. Although the scatter/gather operation can change the contents of TCD Word 7, that operation does not affect the DCHMID*n* registers.

Address: 0x140 + *n**0x1 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7
R	EMI	PAL	0		MID			
W								
Reset	0	0	0	0	0	0	0	0

Figure 194. Channel *n* Master ID Register (eDMA_DCHMID*n*)

Table 273. eDMA_DCHMIDn field descriptions

Field	Description
0 EMI	Enable Master ID replication 0 Master ID replication is disabled. 1 Master ID replication is enabled.
1 PAL	Privileged Access Level The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCD Word 7. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
4:7 MID	Master ID DMA's master ID when channel *n* is active and master ID replication is enabled. **Note:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCD Word 7.

23.3.18 TCD Source Address (eDMA_TCDn_SADDR)

Refer to *Figure 195* and *Table 274* for details.

Address: 0x1000 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SADDR															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SADDR															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 195. TCD Source Address (eDMA_TCDn_SADDR)

Table 274. eDMA_TCDn_SADDR field descriptions

Field	Description
0:31 SADDR	Source address Memory address pointing to the source data.

23.3.19 TCD Transfer Attributes (eDMA_TCDn_ATTR)

Refer to *Figure 196* and *Table 275* for details.

Address: 0x1004 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMOD					SSIZE			DMOD					DSIZE		
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 196. TCD Transfer Attributes (eDMA_TCD*n*_ATTR)

Table 275. eDMA_TCD*n*_ATTR field descriptions

Field	Description
0:4 SMOD	Source address modulo 0 Source address modulo feature is disabled. Not 0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
5:7 SSIZE	Source data transfer size Using a reserved encoding causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte 110 Reserved 111 Reserved
8:12 DMOD	Destination address modulo See the SMOD definition.
13:15 DSIZE	Destination data transfer size See the SSIZE definition.

23.3.20 TCD Signed Source Address Offset (eDMA_TCD*n*_SOFF)

Refer to *Figure 197* and *Table 276* for details.

Address: 0x1006 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SOFF															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 197. TCD Signed Source Address Offset (eDMA_TCD*n*_SOFF)

1. x = Undefined at reset.

Table 276. eDMA_TCD*n*_SOFF field descriptions

Field	Description
0:15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

23.3.21 TCD Minor Byte Count (Minor Loop Disabled) (eDMA_TCD*n*_NBYTES_MLNO)

Refer to *Figure 198* and *Table 277* for details.

If minor loop mapping is disabled (CR[EMLM] = 0), TCD word 2 is defined as follows.

Address: 0x1008+ *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NBYTES															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 198. TCD Minor Byte Count (Minor Loop Disabled) (eDMA_TCD*n*_NBYTES_MLNO)

Table 277. eDMA_TCD*n*_NBYTES_MLNO field descriptions

Field	Description
0:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **Note:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

23.3.22 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (eDMA_TCD*n*_NBYTES_MLOFFNO)

TCD Word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

Address: 0x1008 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	SMLOE	DMLOE	NBYTES													
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	NBYTES															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 199. TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (eDMA_TCD*n*_NBYTES_MLOFFNO)

Table 278. eDMA_TCD*n*_NBYTES_MLOFFNO field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR. 1 The minor loop offset is applied to the SADDR.
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR. 1 The minor loop offset is applied to the DADDR.
2:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

23.3.23 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (eDMA_TCD*n*_NBYTES_MLOFFYES)

TCD Word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset enabled (SMLOE or DMLOE = 1)

Address: 0x1008+ *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	SMLOE	DMLOE	MLOFF													
Reset[(1)]	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MLOFF						NBYTES									
Reset[(1)]	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 200. TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (eDMA_TCD*n*_NBYTES_MLOFFYES)

Table 279. eDMA_TCD*n*_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR. 1 The minor loop offset is applied to the SADDR.
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR. 1 The minor loop offset is applied to the DADDR.
2:21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

23.3.24 TCD Last Source Address Adjustment (eDMA_TCD*n*_SLAST)

Address: 0x100C + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SLAST															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. = Undefined at reset.

Figure 201. TCD Last Source Address Adjustment (eDMA_TCD*n*_SLAST)

Table 280. eDMA_TCD*n*_SLAST field descriptions

Field	Description
0:31 SLAST	Last source address adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.

23.3.25 TCD Destination Address (eDMA_TCD*n*_DADDR)

See TCD Destination Address register figure and DMA_TCD*n*_DADDR field descriptions table as follows.

Address: 0x1010 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DADDR															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 202. TCD Destination Address (eDMA_TCD*n*_DADDR)

Table 281. eDMA_TCD*n*_DADDR field descriptions

Field	Description
0:31 DADDR	Destination address Memory address pointing to the destination data.

23.3.26 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_CITER_ELINKYES)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) register figure and DMA_TCDn_CITER_ELINKYES field descriptions table as follows.

Address: 0x1014 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ELINK	LINKCH						CITER								
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 203. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_CITER_ELINKYES)

Table 282. eDMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:6 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by these six bits by setting that channel’s TCD*n*_CSR[START] bit.
7:15 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (for example: final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. **Note:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

23.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_CITER_ELINKNO)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) register figure and DMA_TCDn_CITER_ELINKNO field descriptions table as follows.

Address: 0x1014 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	CITER														
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 204. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_CITER_ELINKNO)

Table 283. eDMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel.If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:15 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (for example, final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. **Note:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

23.3.28 TCD Signed Destination Address Offset (eDMA_TCD*n*_DOFF)

See TCD Signed Destination Address Offset register figure and DMA_TCD*n*_DOFF field descriptions table as follows.

Address: 0x1016 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DOFF															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 205. TCD Signed Destination Address Offset (eDMA_TCD*n*_DOFF)

Table 284. eDMA_TCD*n*_DOFF field descriptions

Field	Description
0:15 DOFF	Destination address signed offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

23.3.29 TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD*n*_DLASTSGA)

Address: 0x1018 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DLASTSGA															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DLASTSGA															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset

Figure 206. TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD*n*_DLASTSGA)

Table 285. eDMA_TCD*n*_DLASTSGA field descriptions

Field	Description
0:31 DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCD*n*_CSR[ESG] = 0) then – Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. else – This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, else a configuration error is reported.

23.3.30 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCDn_BITER_ELINKYES)

Address: 0x101C + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ELINK	LINKCH						BITER								
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 207. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCDn_BITER_ELINKYES)

Table 286. eDMA_TCDn_BITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:6 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.
7:15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

23.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_BITER_ELINKNO)

Address: 0x101C + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	BITER														
Reset[1]	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 208. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_BITER_ELINKNO)

Table 287. eDMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

23.3.32 TCD Control and Status (eDMA_TCD*n*_CSR)

See TCD Control and Status register figure and eDMA_TCD*n*_CSR field descriptions table as follows.

Address: 0x101E + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	BWC		MAJORLINKCH						DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 209. TCD Control and Status (eDMA_TCD*n*_CSR)

Table 288. eDMA_TCD*n*_CSR field descriptions

Field	Description
0:1 BWC	Bandwidth control Throttles the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch (XBS). **Note:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00 No eDMA engine stalls 01 Reserved 10 eDMA engine stalls for 4 cycles after each r/w 11 eDMA engine stalls for 8 cycles after each r/w
2:7 MAJORLINKCH	Link channel number If (MAJORELINK = 0) then – No channel-to-channel linking (or chaining) is performed after the major loop counter is exhausted. else – After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit.
8 DONE	Channel done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero; The software clears it, or the hardware when the channel is activated. **Note:** This bit must be cleared to write the MAJORELINK or ESG bits. This bit resets to zero.
9 ACTIVE	Channel active This flag signals the channel is currently in execution. It is set when channel service begins, and the eDMA clears it as the minor loop completes or if any error condition is detected.

Table 288. eDMA_TCD*n*_CSR field descriptions (continued)

Field	Description
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. **Note:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD*n*_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
11 ESG	Enable scatter/gather processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **Note:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD*n*_CSR[DONE] bit is set. 0 The current channel’s TCD is normal format. 1 The current channel’s TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
12 DREQ	Disable request {H,L} 0 The channel’s ERQ bit is not affected. 1 The channel’s ERQ bit is cleared when the major loop is complete.
13 INTHALF	Enable an interrupt when major counter is half complete If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered (aka ping-pong) schemes or other types of data movement where the processor needs an early indication of the transfer’s progress. **Note:** If BITER is set to 1, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
14 INTMAJOR	Enable an interrupt when major iteration count completes If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled 1 The end-of-major loop interrupt is enabled
15 START	Channel start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. This bit resets to zero. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

23.4 Functional description

This section provides a description of the DMA request sources, an overview of the microarchitecture, and functional operation of the eDMA module.

23.4.1 eDMA microarchitecture

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules:

- eDMA engine
  - Address path:

    This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

    When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n* _{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including updates to the final address pointer, reloading the TCD*n* _CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
  - Data path:

    This block implements the bus master read/write datapath. It includes 32 bytes of register storage and the necessary multiplex logic to support any data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output.

    The address and data path modules directly support the two-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase), while the data path module implements the second stage of the pipeline (data phase).
  - Program model/channel arbitration:

    This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus (not shown). The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
  - Control:

    This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-

bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

- Transfer control descriptor memory
  - Memory controller:

    This logic implements the dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus.

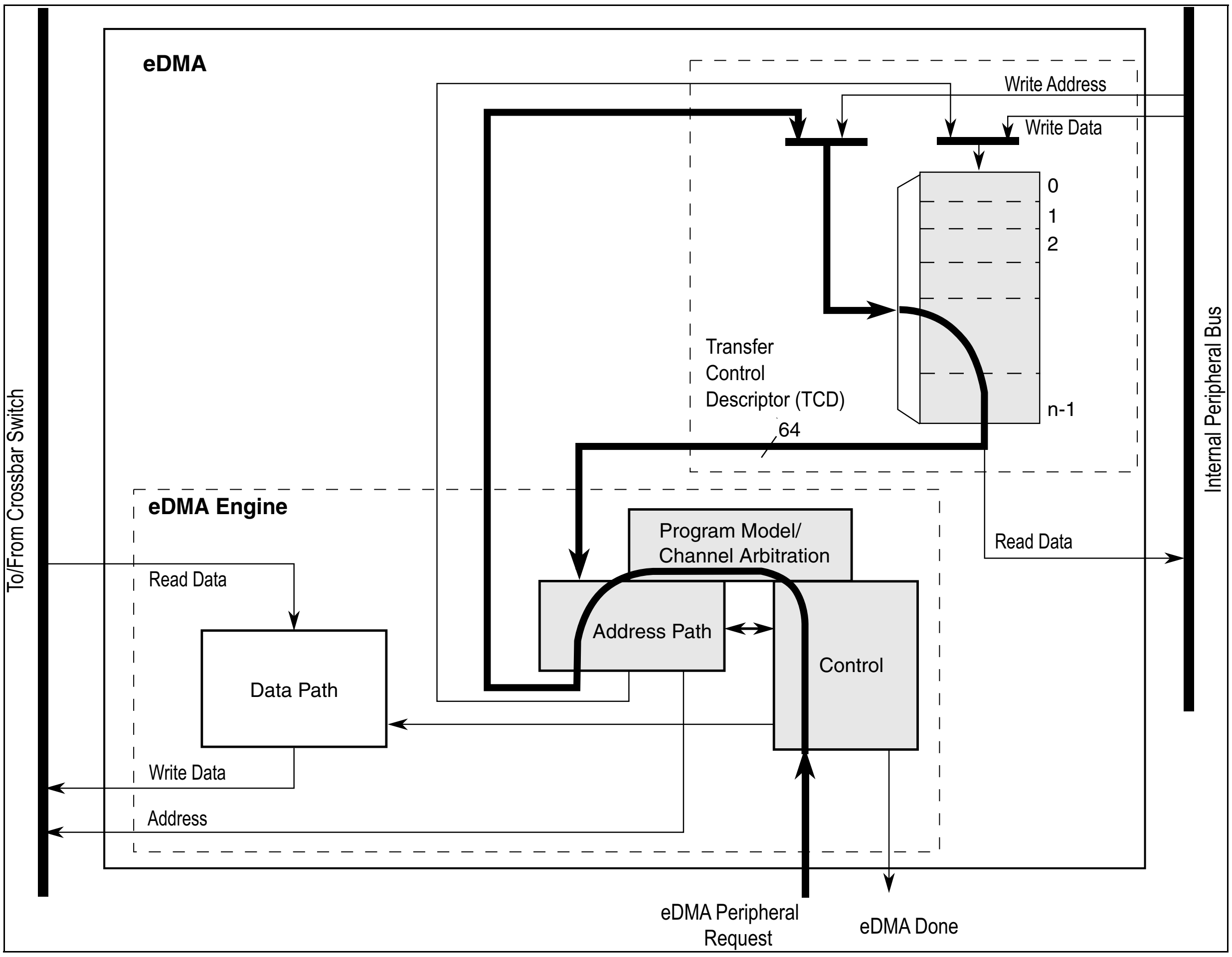
    As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
  - Memory array:

    TCD storage is implemented using a single-port, synchronous RAM array.

23.4.2 eDMA basic data flow

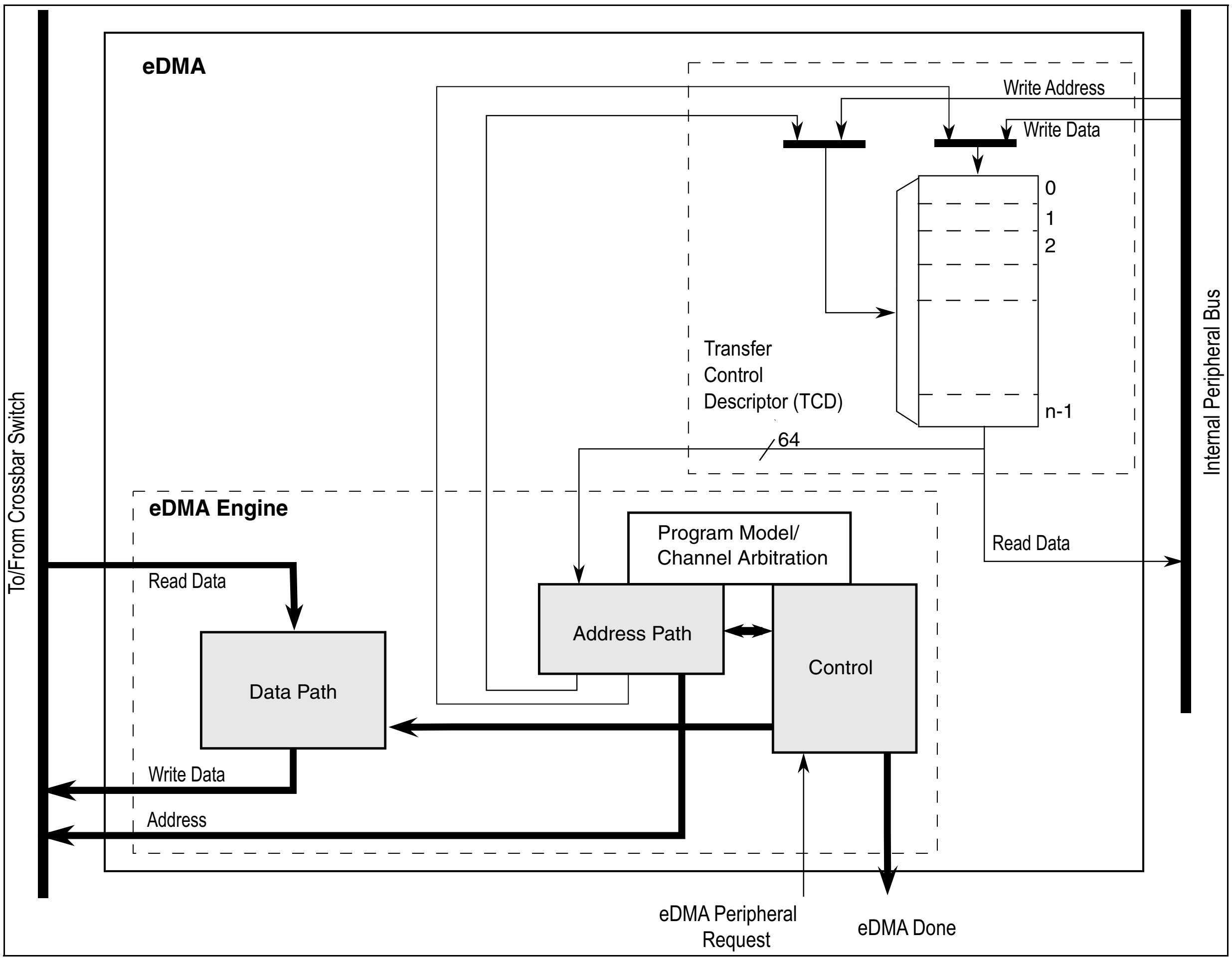
The basic flow of a data transfer can be partitioned into three segments. As shown in *Figure 210*, the first segment involves the channel activation. In the diagram, this example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, channel arbitration is performed, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

Figure 210. DMA operation, part 1



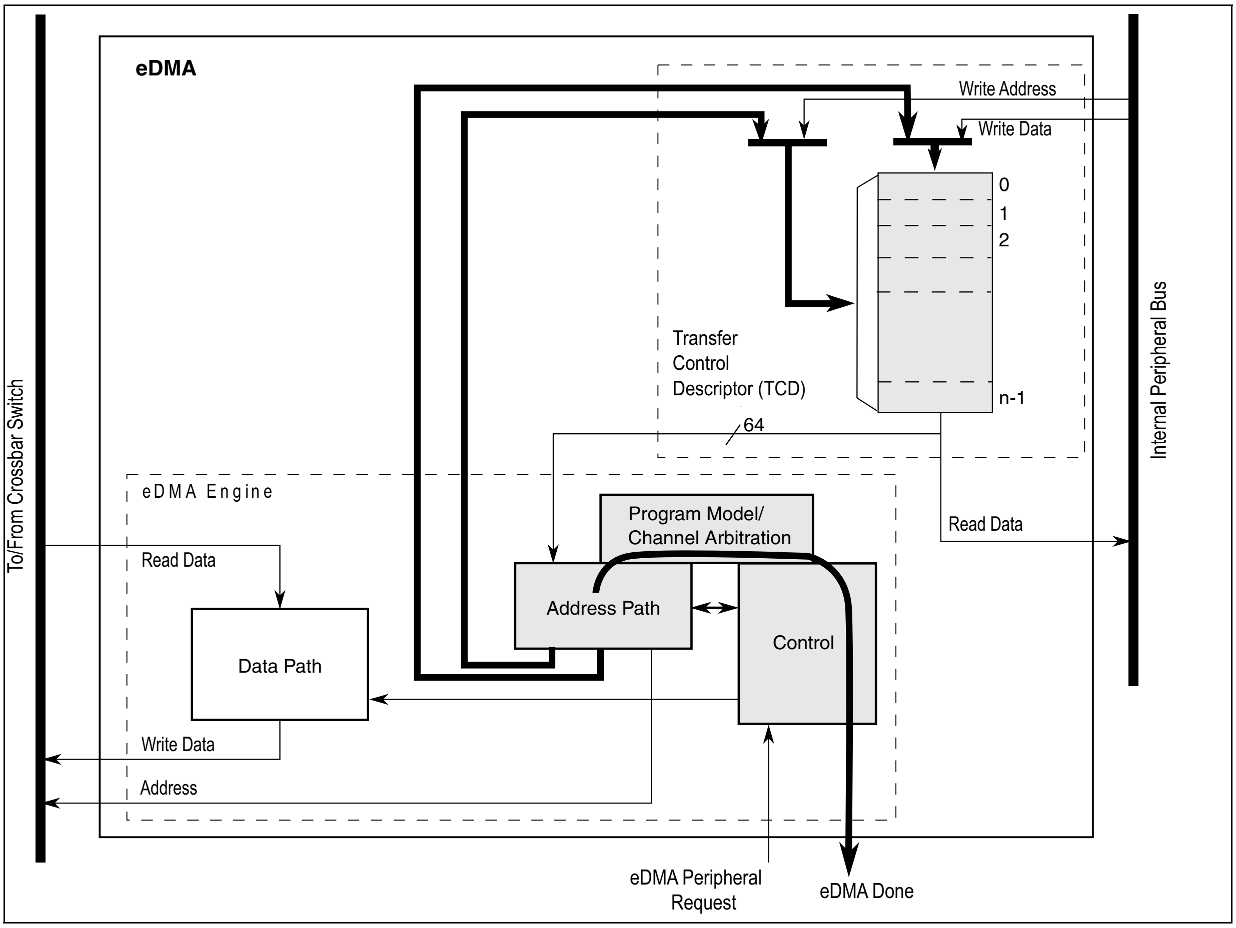
In the second part of the basic data flow as shown in *Figure 211* the modules associated with the data transfer (address path, data path, and control) sequence through source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

Figure 211. DMA operation, part 2



After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, and CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as a possible fetch of a new TCD from memory does using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in *Figure 212*.

Figure 212. DMA operation, part 3



23.4.3 Error reporting and handling

Channel errors are reported in the ES register and can be caused by:

- A configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode),
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle.

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes is detailed in the list below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size, respectively.
- In Fixed Arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups.

All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when Fixed Arbitration mode is enabled.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCD*n*_CITER[E_LINK] bit does not equal the TCD*n*_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system-bus error occurs, the channel terminates after the read or write transaction (which is already pipelined after errant access) has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be canceled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the ES register is updated with the canceled channel number and ECX is set. The TCD of a canceled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD since the aforementioned fields no longer represent the original parameters. When a transfer is canceled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

The occurrence of any error causes the eDMA engine to stop the active channel immediately, and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the ES register. The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

23.4.4 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRI*n*[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes.

A channel's ability to preempt another channel can be disabled by setting DCHPRI*n*[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer; regardless of the lower priority channel's ECP setting. This allows for a pool of low-priority, large data-moving channels to be defined. These low-priority channels can be configured so that they do not preempt each other, thus preventing a low-priority channel from consuming the preempt slot normally available to a true, high-priority channel.

23.4.5 eDMA performance

This section addresses the performance of the eDMA module, focusing on two separate metrics. In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces. In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more interesting metric. In this environment, the speed of the source and destination address spaces remains important, but the microarchitecture of the eDMA also factors significantly into the resulting metric.

The peak transfer rates for several different source and destination transfers are shown in *Table 289* and *Table 290*. The following assumptions apply to those tables:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.
- All internal peripheral bus reads require one wait-state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- All internal peripheral bus accesses are 32 bits in size.
- All internal SRAM accesses are 64 bits in size.

Table 289 presents a peak transfer rate comparison, measured in megabytes per second.

Table 289. eDMA peak transfer rates (Mbyte/s)

System speed, width	From internal SRAM to internal SRAM	From 32-bit internal peripheral bus to internal SRAM	From internal SRAM to 32-bit internal peripheral bus
100 MHz, 64 bits	400.0	160.0	160.0
150 MHz, 64 bits	600.0	240.0	240.0
200 MHz, 64 bits	800.0	320.0	320.0

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, it is assumed the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel. The eDMA design supports the following hardware service request sequence:

- Cycle 1: eDMA peripheral request is asserted.
- Cycle 2: The eDMA peripheral request is registered locally in the eDMA module and qualified. (TCD*n* _CSR[START] bit initiated requests start at this point with the registering of the user write to TCD*n* Word 7).
- Cycle 3: Channel arbitration begins.
- Cycle 4: Channel arbitration completes. The transfer control descriptor local memory read is initiated.
- Cycles 5–6: The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles.
- Cycle 7: The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
- Cycles 8 to *x*[g]: The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.

  The exact timing from this point is a function of the transfer size and response times for the channel's read and write accesses. In this case of two internal peripheral bus reads and a single 64-bit internal SRAM write, the combined data phase time is five cycles. For a single SRAM read and two 64-bit internal peripheral bus writes, the combined data phase time is five cycles.
- Cycle *x*[g] + 1: This cycle represents the data phase of the last destination write.
- Cycle *x*[g] + 2: The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n* Word 7 is read and checked for channel linking or scatter/gather requests.
- Cycle *x*[g] + 3: The appropriate fields in the first part of the TCD*n* are written back into the local memory.
- Cycle *x*[g] + 4: The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
- Cycle *x*[g] + 5: The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every nine cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (three cycles) and SRAM-to-internal peripheral bus (three cycles), DMA requests can be processed every 10 cycles (4 + (3 + 3)/2 + 3). This is the time from Cycle 4 to Cycle *x* + 5. The resulting peak request rate, as a function of the system frequency, is shown in *Table 290*. This metric represents millions of requests per second. A single read/write operation with same transfer size for the source and destination is assumed.

g. "*x*" represents an undefined cycle value dependent on transfer sizes and response times for the channel's read and write accesses.

Table 290. eDMA peak request rate (MReq/s)

System frequency (MHz)	Request rate	
	zero wait states	with wait states
100.0	11.1	10.0
150.0	16.6	15.0
200.0	22.2	20.0

A general formula to compute the peak request rate (with overlapping requests) is:

Equation 1 **PEAKreq = freq / [entry + (1 + read_ws) + (1 + write_ws) + exit]**

where:

- PEAKreq = peak request rate
- freq = system frequency
- entry = channel startup (4 cycles)
- read_ws = wait states seen during the system bus read data phase
- write_ws = wait states seen during the system bus write data phase
- exit = channel shutdown (3 cycles)

For example: consider a system with the following characteristics:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.
- All internal peripheral bus reads require one wait state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- System operates at 150 MHz.

For an SRAM to internal peripheral bus transfer,

Equation 2 **PEAKreq = 150 MHz / [4 + (1 + 0) + (1 + 1) + 3] cycles = 15.0 Mreq/s**

For an internal peripheral bus-to-SRAM transfer,

Equation 3 **PEAKreq = 150 MHz / [4 + (1 + 1) + (1 + 0) + 3] cycles = 15.0 Mreq/s**

The minimum number of cycles to perform a single read/write, with zero wait states on the system bus, from a cold start (where no channel is executing and the eDMA is idle):

- 11 cycles for a software (TCD*n*_CSR[START] bit) request
- 12 cycles for a hardware (eDMA peripheral request signal) request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering are absorbed in or overlap the previous executing channel.

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

23.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

23.5.1 eDMA initialization

A typical initialization of the eDMA has the following sequence:

1. Write the CR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI registers if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service by software (setting the TCD*n*_CSR[START] bit) or hardware (slave device asserting its eDMA peripheral request signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields (shown in *Table 291*) for the selected channel into its internal address path module. As the TCD is read, the first transfer is initiated on the internal bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD*n*_SADDR) to the destination (as defined by the destination address, TCD*n*_DADDR) continue until the specified number of bytes (TCD*n*_NBYTES) are transferred. When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes (interrupts, major loop channel linking, and scatter/gather operations) if enabled.

Table 291. TCD control and status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

Figure 213 shows how each DMA request initiates one minor-loop transfer (iteration) without CPU intervention. DMA arbitration can occur after each minor loop, and one level of

minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

Figure 213. Example of multiple loop iterations

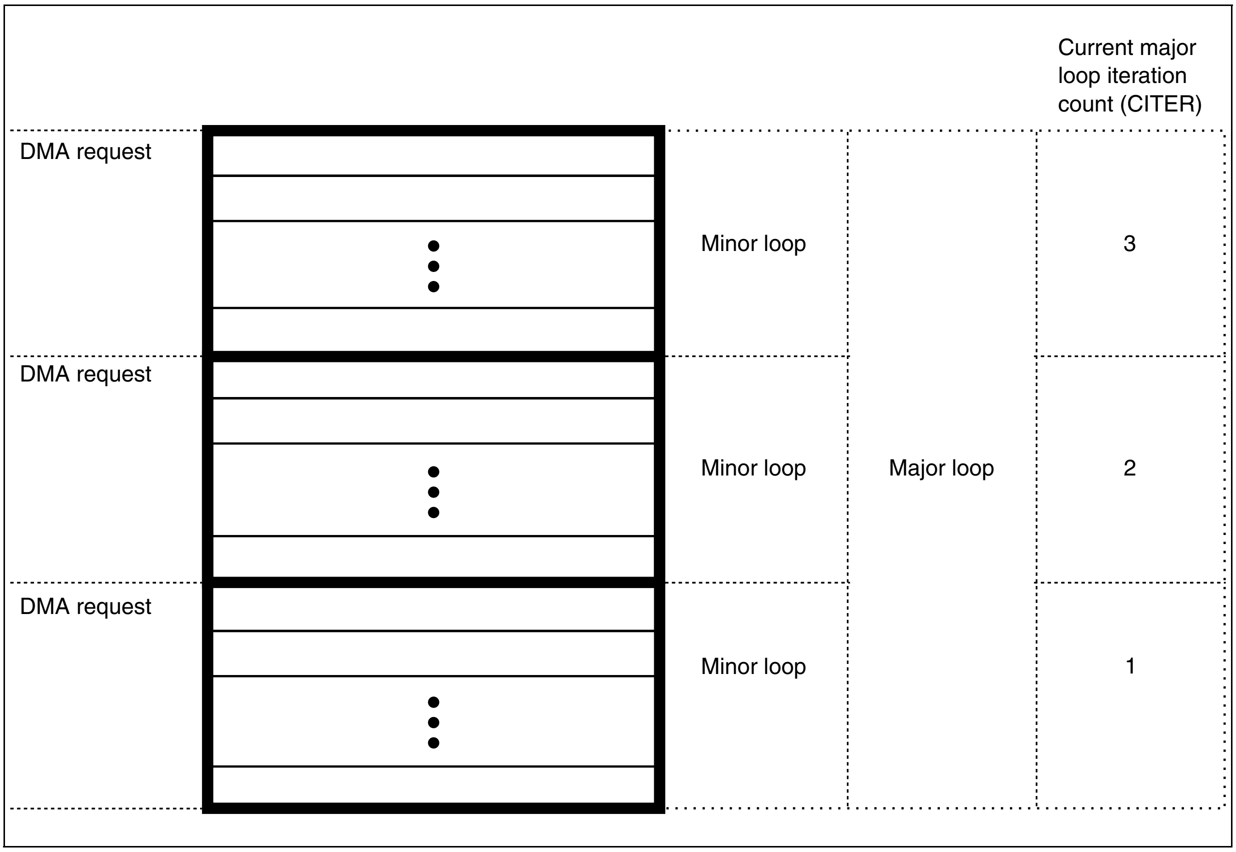
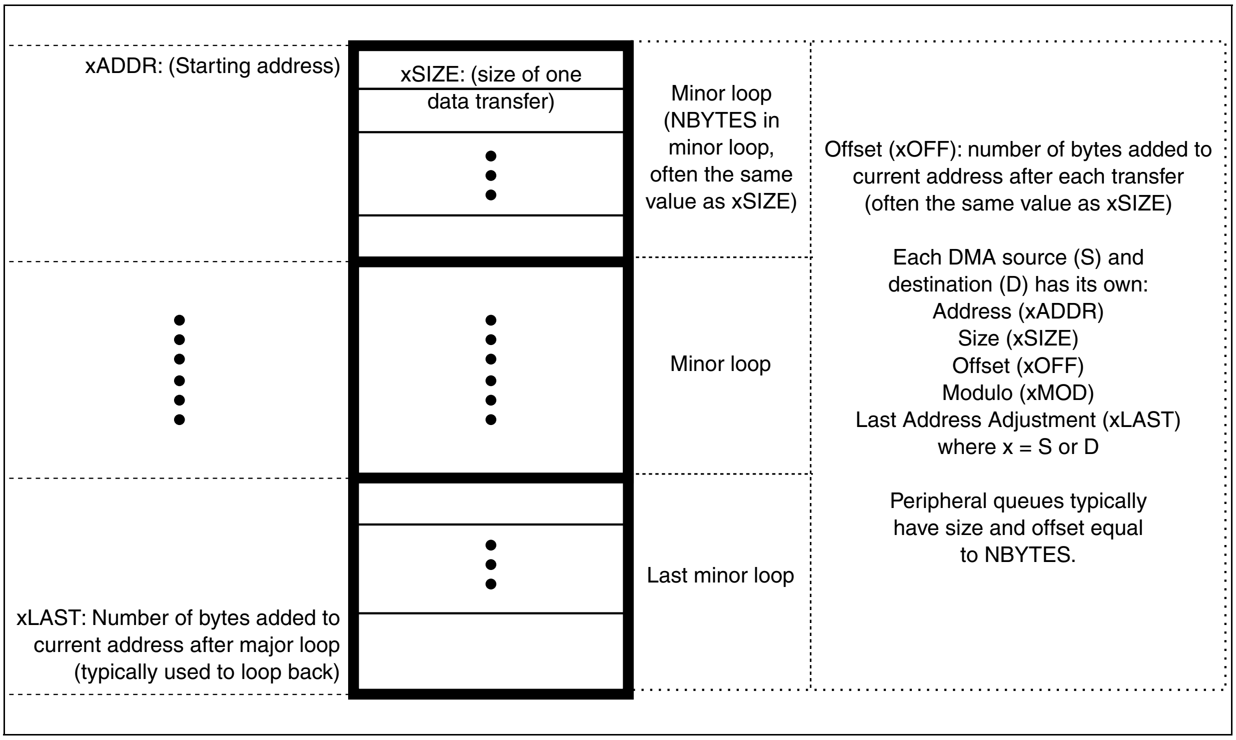


Figure 214 lists the memory array terms and how the TCD settings interrelate.

Figure 214. Memory array terms



23.5.2 DMA programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis with the exception of two errors: the group priority error (ES[GPE]) and the channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the ES register. If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed-group and fixed-channel arbitration modes.
2. Group 3 is the highest priority and all channels are unique in that group.
3. Group 2 is the next highest priority and has two channels with the same priority level.
4. If Group 3 has any service requests, those requests will be executed.
5. Once all of Group 3 requests have completed, Group 2 will be the next active group.
6. If Group 2 has a service request, then an undefined channel in Group 2 will be selected and a channel priority error will occur.
7. This repeats until all of the Group 2 requests have been removed or a higher priority Group 3 request comes in.

In this sequence, for item 2, the eDMA Acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest (channel/group) priority with an active request is selected, but the lowest numbered (channel/group) with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

23.5.3 DMA Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

23.5.3.1 Fixed-group arbitration, fixed-channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so the channels within one group use “fixed” priorities, and that group is assigned the highest “fixed” priority of all groups, it is possible for that group to take all the bandwidth of the eDMA controller—that is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

23.5.3.2 Round-robin group arbitration, fixed-channel arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with a service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round-robin algorithm will select the highest pending request from the next group in the round-robin sequence. Servicing continues round-robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round-robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced. The advantage of this scenario is that no one group will consume all the eDMA bandwidth. The highest priority channel selection latency is potentially greater than fixed/fixed arbitration. Excessive request rates on high priority channels could prevent the servicing of lower priority channels in the same group.

23.5.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in *Section 23.5.3.2: Round-robin group arbitration, fixed-channel arbitration*, but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round-robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round-robin manner, any channel that generates DMA requests faster than a combination of the group round-robin service rate and the channel service rate for its group will not prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high. All channels are treated equally. Priority levels are not used in round-robin/round-robin mode.

23.5.3.4 Fixed-group arbitration, round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in *Section 23.5.3.1: Fixed-group arbitration, fixed-channel arbitration* but all the channels in the highest priority group will get serviced. Service latency will be short on the highest priority group, but could potentially get much longer as the group priority decreases.

23.5.4 DMA transfer

This section discusses how to perform DMA transfers with the eDMA.

23.5.4.1 Single request

To perform a simple transfer of *n* bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop, transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit-wide port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have
 been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a) Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b) Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c) Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d) Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e) Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f) Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g) Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h) Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

23.5.4.2 Multiple requests

Besides transferring 32 bytes via two hardware requests, the next example is the same as the previous one. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware (eDMA peripheral) request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a) Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b) Write 32 bits to location 0x2000 → first iteration of the minor loop.

   c) Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d) Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e) Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f) Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g) Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h) Write 32 bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware (eDMA peripheral) requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
   a) Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
   b) Write 32 bits to location 0x2010 → first iteration of the minor loop.
   c) Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
   d) Write 32 bits to location 0x2014 → second iteration of the minor loop.
   e) Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
   f) Write 32 bits to location 0x2018 → third iteration of the minor loop.
   g) Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
   h) Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete.
14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2(TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

23.5.4.3 Modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address + offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

Table 292 shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567x) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes, and the MOD field is set to 4, allowing for a 16-byte size queue.

Table 292. Modulo feature example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

23.5.5 eDMA TCD*n* status monitoring

This section discusses how to monitor eDMA status.

23.5.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute a sequence for a software activated channel as shown in *Table 293*.

Table 293. TCD status sequence – software activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware (peripheral) initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer’s model.

The TCD status bits execute a sequence for a hardware-activated channel as shown in *Table 294*.

Table 294. TCD status sequence – hardware activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit. The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

23.5.5.2 Active channel TCD*n* reads

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses (SADDR and DADDR) and NBYTES (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

23.5.5.3 Preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in Fixed-Group, Fixed-Channel Arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and group priorities are treated as equal (constantly rotating) when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

23.5.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done → set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a 9-bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note: The TCDn_CITER[E_LINK] bit and the TCDn_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

Table 295 summarizes how a DMA channel can link to another DMA channel (that is, how it can use another channel's TCD) at the end of a loop.

Table 295. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH] Link channel number when linking at end of minor loop (current iteration)	
Link at end of major loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH] Link channel number when linking at end of major loop	

23.5.7 Dynamic programming

23.5.7.1 The following sections describe dynamic programming.

23.5.7.2 Dynamic priority changing

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode.
2. Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing group priority levels:

1. Switch to Round-Robin Group Arbitration mode, change the group priorities, then switch back to Fixed Arbitration mode.
2. Disable all channels, change the group priorities, then enable the appropriate channels.

23.5.7.3 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The coherency model in *Table 296* is recommended when executing a dynamic channel link request.

Table 296. Coherency model for a dynamic channel link request

Step	Action
1	Write 1b to the TCD.major.e_link bit.
2	Read back the TCD.major.e_link bit.
3	Test the TCD.major.e_link request status: – If TCD.major.e_link = 1b, the dynamic link attempt was successful. – If TCD.major.e_link = 0b, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

Note: The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

23.5.7.4 Dynamic scatter/gather

Dynamic scatter/gather is the process of setting the TCD.e_sg bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

Note: The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

23.5.7.4.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model in *Table 297* may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD identification (ID).

Table 297. Coherency model for method 1

Step	Action
1	When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2	Write 1b to theTCD.d_req bit. *Note: Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.*
3	Write theTCD.dlast_sga field with the scatter/gather address.
4	Write 1b to the TCD.e_sg bit.
5	Read back the 16 bit TCD control/status field.
6	Test the TCD.e_sg request status and TCD.major.linkch value: – If e_sg = 1b, the dynamic link attempt was successful. – If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring). – If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

23.5.7.4.2 Method 2 (channel using major loop linking)

For a channel using major loop channel linking, the coherency model in *Table 298* may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 298. Coherency model for method 2

Step	Action
1	Write 1b to theTCD.d_req bit. **Note:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.
2	Write theTCD.dlast_sga field with the scatter/gather address.
3	Write 1b to the TCD.e_sg bit.
4	Read back the TCD.e_sg bit.
5	Test the TCD.e_sg request status: – If e_sg = 1b, the dynamic link attempt was successful. – If e_sg = 0b, read the 32 bit TCD dlast_sga field. – If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring). – If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

24 DMA channel multiplexer (DMACHMUX)

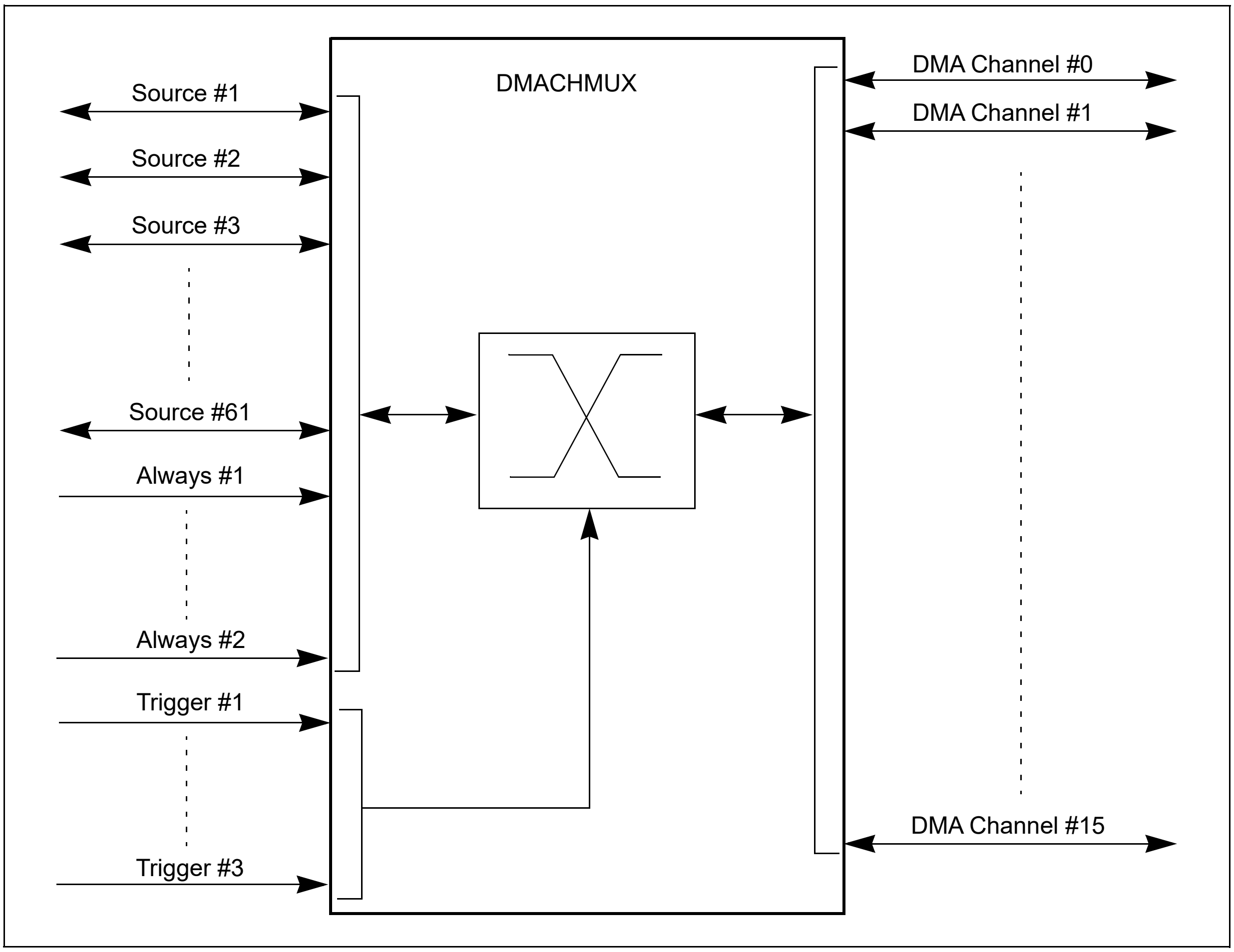
24.1 Introduction

Note: For the chip-specific implementation details of this module's instances, refer to the chip configuration information.

24.1.1 Overview

The DMA channel multiplexer (DMACHMUX) allows to route 61 DMA peripheral sources (called slots) to 16 DMA channels. This is illustrated in *Figure 215*.

Figure 215. DMACHMUX block diagram



24.1.2 Features

The DMACHMUX provides these features.

- 61 peripheral slots + 2 always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channels routers.
  - The first 3 channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of 61 possible peripheral DMA slots or to one of the 2 always-on slots.

24.1.3 Modes of operation

The following operation modes are available.

- Disabled Mode
  In this mode, the DMA channel is disabled. Since disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMACHMUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place (for example, changing the period of a DMA trigger).
- Normal Mode
  In this mode, a DMA source (such as DSPI transmit or DSPI receive for example) is routed directly to the specified DMA channel. The operation of the DMACHMUX in this mode is completely transparent to the system.
- Periodic Trigger Mode
  In this mode, a DMA source may only request a DMA transfer (such as when a transmit buffer becomes empty or a receive buffer becomes full) periodically. Configuration of the period is done in the registers of the Periodic Interrupt Timer (PIT). This mode is only available for channels 0–2 of each DMACHMUX instance.

24.2 External signal description

The DMACHMUX has no external pins.

24.3 Memory map and register definition

24.3.1 Memory map

Table 299 lists the DMACHMUX memory-mapped registers. All addresses are offsets; the absolute address can be computed by adding the specified offset to the base address of the DMACHMUX.

Table 299. DMACHMUX memory map

Address offset	Register	Location
eDMA channel registers per channel (*n* = 0 to 15)		
0x0000 + n*0x1	Channel n Configuration registers (CHCFGn)	*Section 24.3.2.1*

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, CHCFG0 through CHCFG3 are accessible by a 32-bit READ/WRITE to address ‘Base + 0x00’, but performing a 32-bit access to address ‘Base + 0x01’ is illegal.

24.3.2 Register descriptions

This section provides a detailed description of the memory-mapped registers in the DMACHMUX.

24.3.2.1 Channel n Configuration registers (CHCFGn)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMACHMUX slots (peripheral slots or always-on slots) in the system.

Address: 0x0000 + n*0x1 (*n* = 0 to 15) Access: User read/write

	0	1	2	3	4	5	6	7
R/W	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

Figure 216. Channel n Configuration registers (CHCFGn)

Table 300. CHCFGn field descriptions

Field	Description
0 ENBL	DMA Channel Enable ENBL enables the DMA Channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMACHMUX. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel 1 DMA channel is enabled
1 TRIG	DMA Channel Trigger Enable (for triggered channels only) TRIG enables the periodic trigger capability for the DMA Channel. 0 Triggering is disabled. If triggering is disabled, and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel 1 Triggering is enabled
2:7 SOURCE	DMA Channel Source (DMACHMUX slot) SOURCE specifies which DMA source, if any, is routed to a particular DMA channel. Please refer to “Device configuration” chapter for further details about the peripherals and their slot numbers

Table 301. Channel and trigger enabling

ENBL	TRIG	Function	Mode
0	X	DMA Channel is disabled	Disabled Mode

Table 301. Channel and trigger enabling (continued)

ENBL	TRIG	Function	Mode
1	0	DMA Channel is enabled with no triggering (transparent)	Normal Mode
1	1	DMA Channel is enabled with triggering	Periodic Trigger Mode

Note: Setting multiple CHCFG registers with the same source value will result in unpredictable behavior.

Before changing the trigger or source settings a DMA channel must be disabled via the CHCFG[n].ENBL bit.

24.4 DMACHMUX functional description

This section provides a functional description of the DMACHMUX. The primary purpose of the DMACHMUX is to provide flexibility in the system's use of the available DMA channels. As such, configuration of the DMACHMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in *Section 24.5.2: Enabling and configuring sources* is followed, the configuration of the DMACHMUX may be changed during the normal operation of the system.

Functionally, the DMACHMUX channels may be divided into two classes.

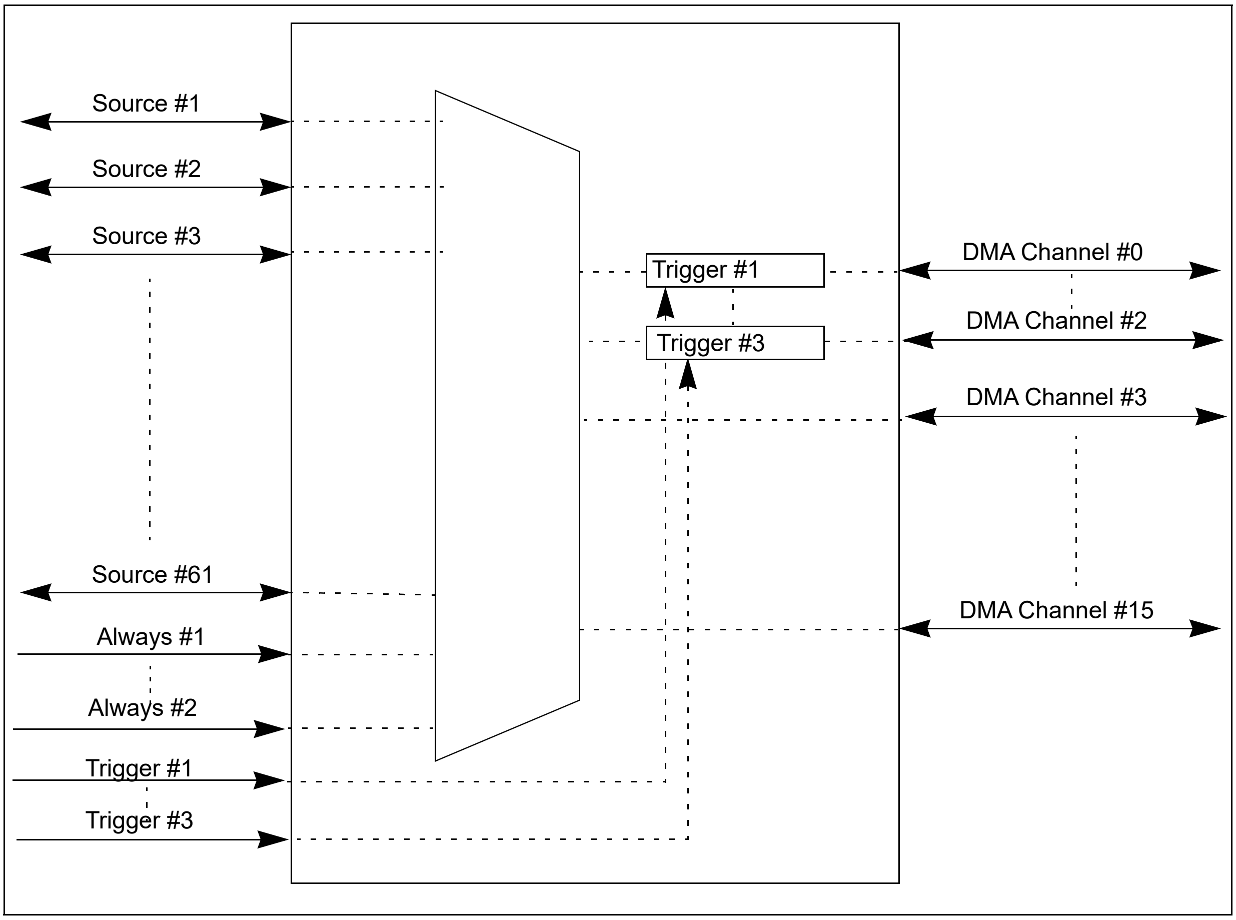
- Channels which implement the normal routing functionality plus periodic triggering capability.
- Channels which implement only the normal routing functionality.

24.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 3 channels of the DMACHMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames or packets at fixed intervals without the need for processor intervention. The trigger is generated by the Periodic Interrupt Timer (PIT): such as the configuration of the periodic triggering interval is done via configuration registers in the PIT. Please refer to the Periodic Interrupt Timer (PIT) chapter for more information on this topic.

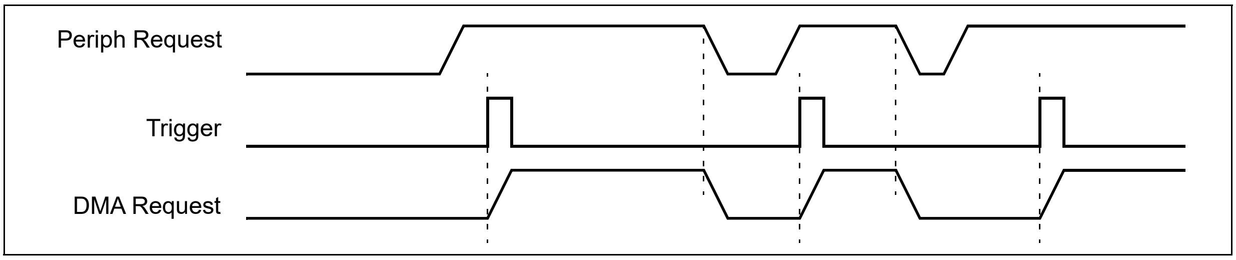
Note: The dynamic nature of the system (such as DMA channel priorities, bus arbitration, interrupt service routine lengths) is such that, the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

Figure 217. DMACHMUX triggered channels



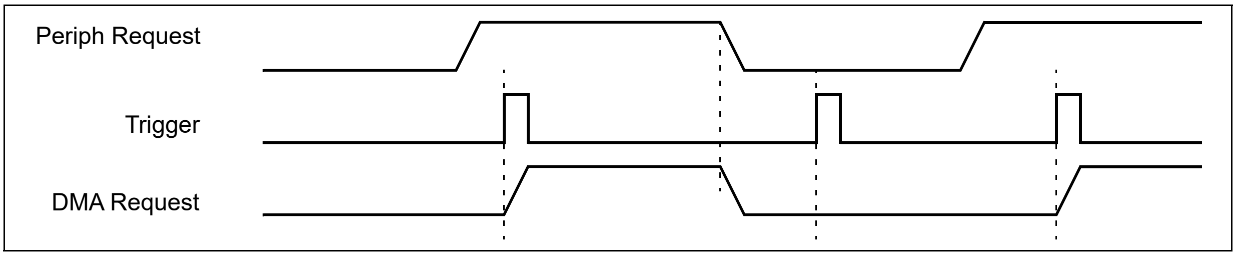
The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the Peripheral to the DMA until a trigger event has been seen. This is illustrated in *Figure 218*.

Figure 218. DMACHMUX channel triggering: normal operation



Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request AND the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, that trigger will be ignored. This situation is illustrated in *Figure 219*.

Figure 219. DMACHMUX channel triggering: ignored trigger



This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations.

- Periodically polling external devices on a particular bus. As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. Once setup, the SPI will request DMA transfers (presumably from memory) as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO Ports to drive or sample waveforms. By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger (such as resolution, range of values) may be found in Periodic Interrupt Timer (PIT) chapter and in the Ethernet chapter.

24.4.2 DMA channels with no triggering capability

The other channels of the DMACHMUX provide the normal routing functionality as described in *Section 24.1.3: Modes of operation*.

24.4.3 “Always Enabled” DMA sources

In addition to the peripherals that can be used as DMA sources, there are 2 additional DMA sources that are “always enabled”. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the *“always enabled”* sources provide no such “throttling” of the data transfers. These sources are most useful in the following cases:

- Doing DMA transfers to/from GPIO: Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Doing DMA transfers from memory to memory: Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Doing DMA transfers from memory to the external bus (or vice-versa): Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation: Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, a *“always enabled”* DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require a new “*start*” event be sent. This can either be a new software activation, or a transfer request from the DMACHMUX. The options for doing this are:

- Transfer all data in a single minor loop. By configuring the DMA to transfer all of the data in a single minor loop (that is major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will incur on the system. For this option, the DMA channel should be disabled in the DMACHMUX.
- Use explicit software reactivation. In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel (by writing to the DMA registers) *after every minor loop*. For this option, the DMA channel should be disabled in the DMACHMUX.
- Use a *“always enabled”* DMA source. In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMACHMUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an *“always enabled”* source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

24.5 Initialization/Application information

24.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (refer to *Section 24.3.2: Register descriptions*). In summary, after reset, all channels are disabled and must be explicitly enabled before use.

24.5.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 2 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

Note: The following is an example. Refer to the chip configuration details for the number of this device’s DMA channels that have triggering capability.

Example 1 Configure source #5 Transmit for use with DMA Channel 2, with periodic triggering capability:

1. Write 0x00 to CHCFG2 (Base Address + 0x02).
2. Configure Channel 2 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG2 (Base Address + 0x02).

The following code example illustrates steps #1 and #4 above:

In file **registers.h:**

```
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
#include "registers.h"
   :
   :
*CHCFG2 = 0x00;
*CHCFG2 = 0xC5;
```

To enable a source without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 3 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

Note: The following is an example. Refer to the chip configuration details for the number of this device's DMA channels that have triggering capability.

Example 2 Configure source #5 Transmit for use with DMA Channel 2, with <u>no</u> periodic triggering capability:

1. Write 0x00 to CHCFG2 (Base Address + 0x02).
2. Configure Channel 2 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG2 (Base Address + 0x02).

The following code example illustrates steps #1 and #3 above:

In File **registers.h:**

```
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
```

```
  volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
  volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
  volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
  #include "registers.h"
     :
     :
  *CHCFG2 = 0x00;
  *CHCFG2 = 0x85;
```

Disabling a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module specific configuration may be necessary. Please refer to the appropriate section for more details.

Switching the source of a DMA Channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the ENBL and TRIG bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the ENBL and TRIG bits are set.

Example 3 Switch DMA Channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 does not have triggering capability.
2. Write 0x00 to CHCFG8 (Base Address + 0x08).
3. Write 0x87 to CHCFG8 (Base Address + 0x08). (In this example, setting the TRIG bit would have no effect, due to the assumption that channels 8 does not support the periodic triggering functionality).

The following code example illustrates steps #2 and #3 above.

In File **registers.h:**

```
  #define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
  /* Following example assumes char is 8-bits */
  volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
  volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
  volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
  volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
  volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
```

```
  volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
  volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
  volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
  volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
  volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
  volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
  volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
  volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
  volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
  volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
  volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
  #include "registers.h"
     :
     :
  *CHCFG8 = 0x00;
  *CHCFG8 = 0x87;
```

25 Clocking

25.1 Introduction

This chapter describes the architecture of the system level clocks and includes the following information:

- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers
- Progressive Clock Switching (PCS)

Note: This chapter only covers clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.

The system clock (SYS_CLK) services the computational shell module (includes the main CPU Main Core_2), that run in the high-speed domain, memories and debug logic. Its independence from the peripheral clocks allows SYS_CLK to be a frequency modulated (FM) clock with reduced electromagnetic emissions while having a precise clock for the peripheral timer and communication functions. It can be reduced during low computational activity periods to lower power consumption while providing a fixed frequency to the communication interfaces and timers.

The internal 16 MHz RC oscillator (IRCOSC) services device boot up and may be enabled via the Mode Entry Module in the event of PLL or external oscillator failure. See the “Mode Entry Module (MC_ME)” chapter for details.

The alternative clock sources are:

- External oscillator/crystal (XOSC)
- Internal 16 MHz RC oscillator (IRCOSC)
- PLL0 (PLL0_PHI)
- PLL1 (PLL1_PHI)

Auxiliary Clock Selectors in the Clock Generation Module (MC_CGM) allow developers to select an independent clock source for each system peripheral. There is an additional System Clock Selector used exclusively for the system clock.

The outputs of the module clock selectors have individual dividers that can divide the clock selector outputs by up to 512 for a given peripheral.

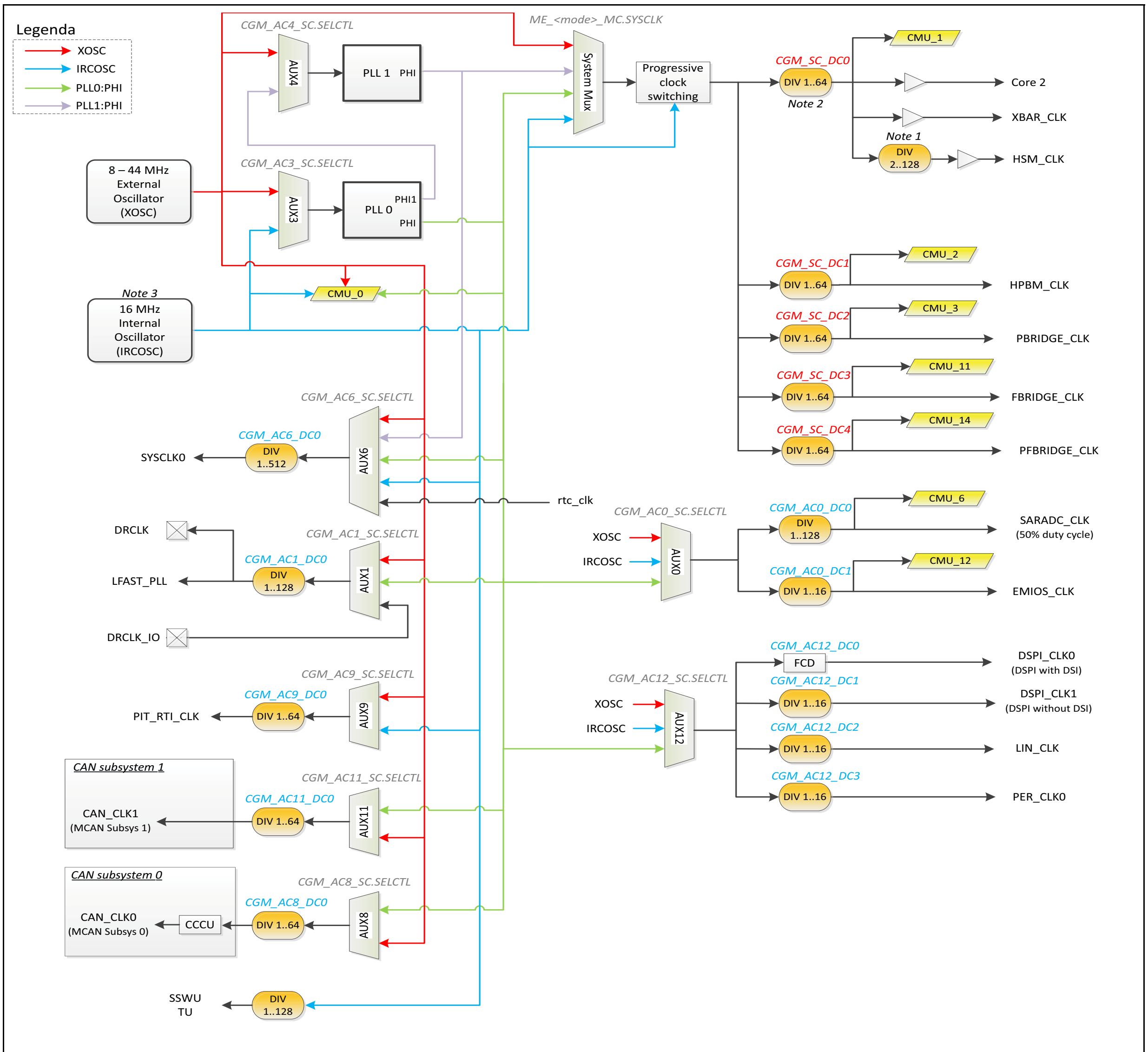
Clock Monitor Units (CMUs) are connected to outputs of clock dividers to verify that clock frequencies stay within operating limits and can be configured to signal an interrupt or initiate a system reset when an issue with a clock is found.

25.2 Clock generation

The top-level clock generation architecture can be seen in *Figure 220*. See *Chapter 28: Clock Generation Module (MC_CGM)*for details on the Fractional Clock Dividers. All clock selectors and dividers are located and programmed in the MC_CGM, with the exception of the Hardware Security Module (HSM) clock divider. All dividers are integer dividers (1, 2, 3, …, *n*) with the ranges shown in the divider blocks in *Figure 220*. All dividers connected to the System Clock Selector must have 50% duty cycle outputs.

Note: Clock Monitoring Units CMU1 to CMU14 in Figure 220 use IRCOSC as their reference clock. The user can select either IRCOSC or XOSC as the reference clock for CMU_0.

Figure 220. Clock generation

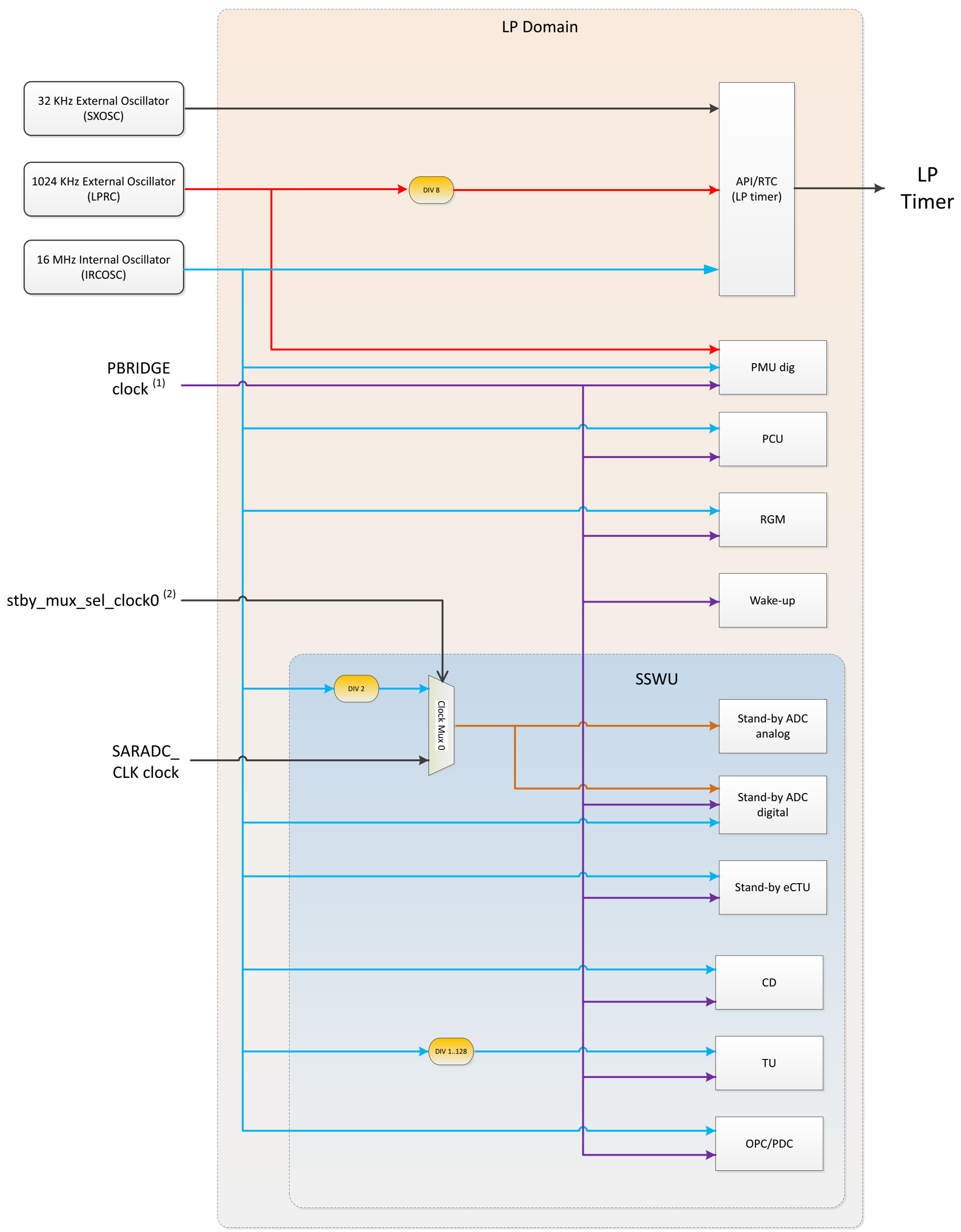


All dividers shown in the diagram are integer dividers with a range of 1, 2, 3,…., n
FCD are fractional clock dividers.
All clock dividers are 50% duty cycle.
Note 1: located within the CSM memory space.It shall be set only to 2 or 4

Note 2: CGM_SC_DC0 divisor can only be set to 1
CGM_SC_DC1 divisor can only be set to 2;
CGM_SC_DC2 divisor can only be set to 4;
Note 3: within the stand-by domain

25.2.1 STAND-BY Domain

Figure 221. Clock distribution in Stand-by Domain



Note 1: IP registers clock, gated for each IP in main domain. not running during stand-by.
Note 2: Default selection on IRC 16MHz. To be set when entering in stand -by.Selection of alternate clock from main domain when stand-by is not used.

ADC 10-bit conversion frequency in smart stand-by: 8MHz (IRCOSC 16 MHz divided by 2). This frequency guarantees the ADC precision required during stand-by.

IRCOSC clock is used to clock all the other IPs used for Smart-stand-by (SSWU). It can be used also to clock PMC during stand-by.

Divider for RCOSC vs ADC and stand-by peripherals can be fixed or programmable (suggested ratios: 1, 2, 4).

IRCOSC can be used without fine calibration during Smart stand-by, if Tmax during stand-by is less than 55°C.

IRCOSC hard macro can be put in power down (see “Smart Stby Gating” in *Figure 221*) when not necessary to clock IPs for Smart-stand-by. This adds a 5 µs delay due to IRCOSC wake up to the smart stand-by execution time, but this delay is acceptable. IRCOSC will be woken up by an RTC/API trigger and will be put in power down by the SSWU Stand-by CTU end of sequence signaling.

Upon entering the stand-by domain, the SW shall switch both the PLL off, because they will not be switched on when the stand-by mode is exited. Rather, the SW shall turn them on again when exiting from the stand-by mode.

25.2.2 MC_CGM registers

Table 302 shows the relationship between the clocks in *Figure 220* and the MC_CGM registers.

Table 302. MC_CGM relationship to clocks

MC_CGM register(s)	Register description	Output clock	CMU(1)
CGM_SC_DC0	System Clock Divider Configuration 0	CORE_CLK (Core_2) XBAR_CLK HSM_CLK	CMU_1_CORE_XBAR
CGM_SC_DC1	System Clock Divider Configuration 1	HPBM_CLK (DMA1, SIPI/1, SWT, STM, INTC)	CMU_2_HPBM
CGM_SC_DC2	System Clock Divider Configuration 2	PBRIDGE_CLK (All peripheral bridges and peripherals not connected to any other system clock divider)	CMU_3_PBRIDGE
CGM_SC_DC3	System Clock Divider Configuration 3	FBRIDGE_CLK	CMU_11_FBRIDGE
CGM_SC_DC4	System Clock Divider Configuration 4	PFBRIDGE_CLK (M_CANx)	CMU_14_PFBRIDGE
CGM_AC0_DC0 CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	SARADC_CLK	CMU_6_SARADC
CGM_AC0_DC1 CGM_AC0_SC	Aux Clock 0 Divider Configuration 1 Aux Clock 0 Select Control	EMIOS_CLK	CMU_12_EMIOS
CGM_AC1_DC0 CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	LF_REF_CLK (PD LFAST PLL, DRCLK)	—
CGM_AC3_SC	Aux Clock 3 Select Control	PLL0	—
CGM_AC4_SC	Aux Clock 4 Select Control	PLL1	—

Table 302. MC_CGM relationship to clocks (continued)

MC_CGM register(s)	Register description	Output clock	CMU[1]
CGM_AC6_DC0 CGM_AC6_SC	Aux Clock 6 Divider Configuration 0 Aux Clock 6 Select Control	SYSCLK0	—
CGM_AC8_DC0 CGM_AC8_SC	Aux Clock 8 Divider Configuration 0 Aux Clock 8 Select Control	CAN_CLK0	—
CGM_AC9_DC0 CGM_AC9_SC	Aux Clock 9 Divider Configuration 0 Aux Clock 9 Select Control	RTI_CLK	—
CGM_AC11_DC0 CGM_AC11_SC	Aux Clock 11 Divider Configuration 0 Aux Clock 11 Select Control	CAN_CLK1	—
CGM_AC12_DC0 CGM_AC12_SC	Aux Clock 12 Divider Configuration 0 Aux Clock 12 Select Control	DSPI_CLK0 (DSPI4/5)	—
CGM_AC12_DC1 CGM_AC12_SC	Aux Clock 12 Divider Configuration 1 Aux Clock 12 Select Control	DSPI_CLK1 (DSPI0/1/2/3/6)	—
CGM_AC12_DC2 CGM_AC12_SC	Aux Clock 12 Divider Configuration 2 Aux Clock 12 Select Control	LIN_CLK (LIN0/1/2/3/4/5/6/7/8/9/10/11/12/15)	—
CGM_AC12_DC3 CGM_AC12_SC	Aux Clock 12 Divider Configuration 3 Aux Clock 12 Select Control	PER_CLK0	—
Note: See the Clock Generation Module (MC_CGM) for specifics on register implementation.			

1. CMU_0 is not shown in the table since it does not monitor an output clock from the MC_CGM.

25.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in *Table 303*.

Table 303. Maximum system level clock frequencies

Clock Selector	Nominal programmed max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
System clock selector[1]	120	0	CORE_CLK (Core_2), XBAR_CLK	120
System clock selector[1]	120	1	HPBM_CLK	60
System clock selector[1]	120	2	PBRIDGE_CLK	30
System clock selector[1]	120	3	FBRIDGE_CLK	120

Table 303. Maximum system level clock frequencies (continued)

Clock Selector	Nominal programmed max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
System clock selector(1)	120	4	PFBRIDGE_CLK	120
System clock selector(1)	120	—	HSM_CLK (2)	60
Note: The user is responsible to verify that these values are not exceeded. **Note:** Refer to *Figure 220* for clocking architecture overview.				

1. System clock selection is a component of the device run mode switching process. Refer to the Mode Entry Module (MC_ME) chapter for details.
2. Maximum frequency is related to HSM core clock (HSM_CLK). Refer to *Figure 223* for details.

Table 304. Maximum auxiliary level clock frequencies

Clock Selector	Nominal programmed max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 0 Selector	400	0	SARADC_CLK	16
Aux Clock 0 Selector	400	1	EMIOS_CLK(1)	50
Aux Clock 3 Selector	Refer to the Data Sheet	—	PLL0	—
Aux Clock 4 Selector	Refer to the Data Sheet	—	PLL1	—
Aux Clock 6 Selector	400	0	SYSCLK0	Refer to the Data Sheet, section “I/O output DC characteristics”
Aux Clock 8 Selector	400	0	CAN_CLK0	If CCCU is enabled: 400 If CCCU is bypassed: 80
Aux Clock 9 Selector	40	0	RTI_CLK	40
Aux Clock 11 Selector	400	0	CAN_CLK1	80
Aux Clock 12 Selector	400	0	DSPI_CLK0(2)	100
Aux Clock 12 Selector	400	1	DSPI_CLK1(2)	100
Aux Clock 12 Selector	400	2	LIN_CLK0	100

Table 304. Maximum auxiliary level clock frequencies (continued)

Clock Selector	Nominal programmed max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 12 Selector	400	3	PER_CLK0	80
Note: The user is responsible to verify that these values are not exceeded. **Note:** Refer to the *Figure 220* for clocking architecture overview.				

1. When BCTU is used to route a trigger event to SARADC, its clock (EMIOS_CLK) must be greater than SARADC_CLK.
2. DSPI_CLK0/1 frequency should always be greater than or equal to 1.25 times of PBRIDGE_CLK.

In order to maintain synchronization between the different system clock branches (for example, SYS_CLK output from the System Clock Selector), there are limitations on the frequencies of those clocks. The system clocks must be binary multiples of each other with the frequency relationships described in the electrical specifications.

25.3.1 JTAG frequencies

Table 305 shows the maximum frequencies of the JTAG interface.

Table 305. JTAG frequencies

Configuration	e200z4 max. internal JTAG frequency	Max. external JTAG frequency	Notes
Unconfigured clock	—	Refer to electrical characteristics for maximum usable external JTAG frequency	Assumes 16 MHz clock on PD (IRC) with +/- 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
Clock configured to 120 MHz	60 MHz		JTAG usable at 1/2 CPU frequency

25.4 Default clock configuration

At power up, the 16 MHz IRC is the default system clock for the chip. All clock source and clock control blocks should be reset to their default state.

All auxiliary clock selectors will have a default selection of the first available clock source mapped on the respective clock mux. All clock dividers attached to Auxiliary clock selectors (that is not the System clock selectors) are disabled by default.

At power up, all system clock (SYS_CLK branch) dividers are set to 1 to allow a fast DCF reading operation. From the end of Phase 3, all system clock (SYS_CLK branch) dividers are set to divide by a ratio as described in *Table 306*. The SW shall be able to change them accordingly with the divisor range (keeping in account the clock constraints).

Table 306. SYS_CLK divisor clock ratios at power up

Divisor	Description	Ratio
CGM_SC_DC0	cores, XBAR	1
CGM_SC_DC1	high speed bridge	2
CGM_SC_DC2	slow peripheral bridge	4
CGM_SC_DC3	fast peripheral bridge	1
CGM_SC_DC4	slow-fast peripheral bridge	4

Software must have the option to switch the system clock between the available clocks (PLL, XTAL, RCOSC) and must not cause a glitch.

Under the software responsibility, the value chosen for the divisors CGM_SC_DC0, CGM_SC_DC1 and CGM_SC_DC2 shall respect the ratios shown in *Table 306*.

Synchronous IP blocks need to be clocked during reset for proper initialization. After reset, all clock gates should be disabled except for the clocks required for Core_2 execution. The internal RC oscillator is always kept on and the output clock is considered as safe. This safe clock must be used in the event of PLL or oscillator failure.

25.5 Clock sources

The following clock sources are described in this section:

- PLL0
- PLL1 (SSCG - Spread Spectrum Clock Generator (FM))
- External oscillator (XOSC)
- External Clock (EXTAL Bypass)
- 16 MHz internal RC oscillator (IRCOSC)

IRCOSC is also the system clock available during the STANDBY and it is placed in STANDBY power domain.

Additionally, there are two low power oscillators in STANDBY domain:

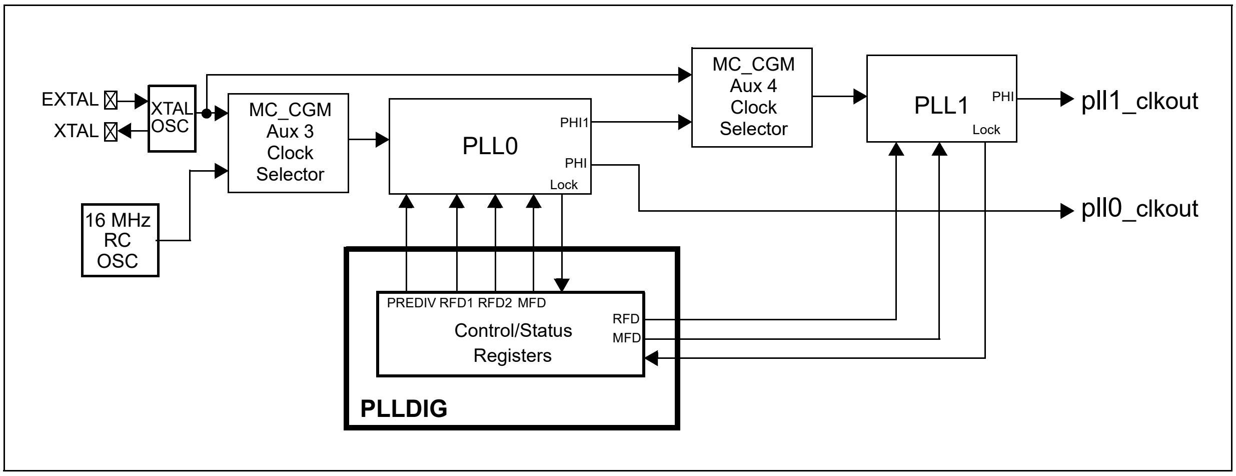
- Low power RC oscillator 1024 KHz (LPRC)
- Slow external crystal oscillator 32 KHz (SXOSC)

These two low power oscillators can be used as clock for counters of API/RTC.

25.5.1 PLL

The Dual PLL provides separate system and peripheral clocks as shown in *Figure 220*. The PLLs are disabled after power is cycled and must be enabled by software. The block diagram for the Dual PLL is shown in *Figure 222*.

Figure 222. SPC584Bx PLL digital interface block diagram



25.5.1.1 PLL0: base PLL (non-FM)

This primary PLL provides a clock that is not Frequency Modulated to the peripheral IPs and the reference clock to PLL1.

The input clock sources for PLL0 are:

- XOSC
- IRCOSC
- The external reference clock (input on the EXTAL pin)

The output clocks from PLL0 are:

- PHI: drives the peripheral clock and the system clock when PLL1 is not locked or active.
- PHI1: provides the input reference clock for PLL1.

25.5.1.2 PLL1: FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is used to drive the system clock.

The input clocks to PLL1 are:

- PHI1 output clock from PLL0
- The external reference clock.

The output clock from PLL1 is:

PHI: drives the system clock. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

25.5.1.3 PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of the PLLs. Details are provided in the “Dual PLL Digital Interface” chapter.

25.5.1.4 PLL register write protection

The PLL registers defined in *Table 307* have write protection with:

- Soft locking: can be unlocked by software after being previously locked.
- Hard locking: can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the PLL register write protection (refer to *Section 7.10.4: Register protection (REG_PROT) configuration* in *Chapter 7: Device configuration* for details).

Table 307. PLL register write protection

Offset	Register
0x00	PLL0 Control Register (PLL0CR)
0x04	PLL0 Status Register (PLL0SR)
0x08	PLL0 Divider Register (PLL0DIVR)
0x20	PLL1 Control Register (PLL1CR)
0x24	PLL1 Status Register (PLL1SR)
0x28	PLL1 Divider Register (PLL1DIVR)
0x2C	PLL1 Frequency Modulation Register (PLL1FMR)
0x30	PLL1 Fractional Divider Register (PLL1FDR)
Note: Refer to Register Protection Configuration chapter for bit field details.	

25.5.1.5 PLL register reset values

All reset values for the PLLDIG registers that are device specific are shown in *Table 308*. All other register reset values are shown in the Dual PLL Digital Interface chapter.

Table 308. Dual PLL Digital Interface register reset values

Offset (hex)	Register	Reset value (hex)
0x0008	PLL0 Divider Register (PLL0DV)	0x0000_0000
0x0028	PLL1 Divider Register (PLL1DV)	0x0000_0000
0x0034	Reserved	—
0x0038	Reserved	—

25.5.1.6 Loss of clock detection

Loss of clock detection should be enabled at all times when the PLLs are enabled. There is a software option to generate a reset or interrupt in the event of a loss of clock condition.

25.5.1.7 PLLDIG initialization information

Configure PLL0 and related modules.

1. With PLL0 disabled, program PLL0 clock source in MC_CGM_AC3_SC[SELCTL]. Default source is 16 MHz IRCOSC. Write MC_CGM_AC3_SC[SELCTL] = 1 to select XOSC as source.
2. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].
3. If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the MC_ME).
4. If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.

Turn on XOSC and PLL0.

5. Configure a mode configuration for turning on PLL0 and XOSC. In a mode configuration register (for example, MC_ME_DRUN_MC) set MC_ME_<mode>_MC[XOSCON] = 1 and MC_ME_<mode>_MC[PLL0ON] = 1. If desired, also set MC_ME_<mode>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0(PLL0 PHI output) as the sysclk.
6. Enter that mode by two writes to MC_ME_CTRL register to enter that mode. This is required even if entering the same mode.

Wait for the mode transition to complete.

7. Wait for the mode transition to complete by polling MC_ME_GS[S_MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

   XOSC counter expires (if the new mode configuration changes MC_ME_<mode>_MC[XOSCON] = 1), and

   PLL is locked (if the new mode configuration changes MC_ME_<mode>_MC[PLL0ON] = 1)
8. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

Configure PLL1 and related modules.

9. With PLL1 disabled, program PLL1 clock source in MC_CGM_AC4_SC[SELCTL]. Default is MC_CGM_AC4_SC[SELCTL] = 1 which selects XOSC.
10. Program PLL1DV[MFD] and PLL1DV[RFDPHI].
11. If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.

Turn on PLL1.

12. Configure a mode configuration for turning on PLL1, and keeping XOSC and PLL1 on. In a mode configuration register (for example, MC_ME_DRUN_MC), initialize MC_ME_<mode>_MC[XOSCON] = 1, MC_ME_<mode>_MC[PLL1ON] = 1 and MC_ME_<mode>_MC[PLL1ON] = 1. If desired, also set MC_ME_<mode>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.
13. Enter that mode by two writes to ME_CTRL register to enter that mode. This is required even if entering the same mode.

Wait for the mode transition to complete.

14. Wait for the mode transition to complete by polling MC_ME_GS[S_TRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

    XOSC counter expires (if the new mode configuration changes XOSCON to 1), and

    PLL1 is locked (if the new mode configuration changes MC_ME_<mode>_MC[PLL1ON] = 1).

15. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

Note: Refer to the "Mode Entry Module (MC_ME)" chapter in this Reference Manual for more details.

25.5.2 External oscillator (XOSC)

The external oscillator (XOSC):

- Is a reference for the on-chip PLLs.
- Can be used as a clock source for the ADCs, timer, serial interfaces, RTI, LFAST and as a system clock source.
- Is available for observation on either of the SYSCLK0/1 pins.
- Is used as a reference to calibrate the IRCOSC frequency.

The external oscillator allows a crystal or external clock to be used as the reference clock for the microcontroller. The XOSC has the following features:

- Internal load capacitors for 20 MHz/40 MHz crystal oscillators
- Automatic Loop Control (ALC) to remove need for external series resistor
- Gain selection for <= 20 MHz and 40 MHz crystals to allow for optimal startup margin without overdrive issues
- Reference clock to PLL0
- Reference clock to CMU0 (IRCOSC trimming)
- Option to drive the CAN protocol clocks directly from the XOSC

The XOSC provides support for 4 MHz - 40 MHz crystal inputs, has integrated load capacitors and ALC to remove the need for external series resistor

25.5.2.1 XOSC Start-up

Enabling of the oscillator at startup is determined by a bit of Miscellaneous DCF in UTEST Flash memory (refer to *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*). The default value has the XOSC disabled:

UTEST Miscellaneous[24] = 0 (XOSC_EN).

If enabled, the oscillator is started in PHASE 3 of the reset sequence: after negation of the internal POR circuits but while the external PORST input is asserted. This is done to allow fast application start-up time while accounting for settling time of the power regulators.

The selection of internal or external load capacitors on the XTAL/EXTAL pins is also determined by a bit in the UTEST row of flash memory. The default value is external load capacitors:

UTEST Miscellaneous[25] = 1 (XOSC_EXT_LOAD).

Load capacitor values are determined by the crystal manufacturer data sheet requirements, while accounting for stray PCB and on-chip capacitance. Refer to the device data sheet for on-chip capacitance values.

When using the internal load capacitors for the oscillator, the startup value of the capacitors is stored in the UTEST row of the flash memory. During PHASE 3 of the device reset sequence, the oscillator UTEST values are read and driven to the oscillator. The enable/disable state of the oscillator is captured in the MC_ME module at this time. After reset, the oscillator can be enabled/disabled by software in the MC_ME module. Internal load capacitor selection is maintained by the oscillator if disabled after reset. The default reset value for the XOSC trimming capacitors does not trim the internal capacitor values:

UTEST Miscellaneous[26:30] = 00000b (XOSC_LOAD_CAP_SEL).

The internal load capacitor values stored in UTEST flash memory row have triple-voting flip-flop (TVF) implementation to prevent an incorrect value causing an undetectable error in the system.

The XOSC has the ability to be started with 4 MHz - 40 MHz crystal. The default value of this field is for an 35 MHz - 40 MHz:

UTEST Miscellaneous[19:21] = 111b (XOSC_FREQ_SEL).

Refer to the FLASH memory chapter for more information on the UTEST row programming.

25.5.2.2 XOSC register write protection

The XOSC registers defined in *Table 309* have write protection with:

- Soft locking: can be unlocked by software after being previously locked.
- Hard locking: can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the XOSC register write protection.

Table 309. XOSC register write protection

Offset	Register	Protections
0x00	XOSC Control Register (XOSC_CTL)	Yes
Note: refer to “Register Protection Configuration” chapter for bit field details		

25.5.2.3 XOSC reset value

Table 310 shows the default reset value for the XOSC registers.

Table 310. XOSC register reset values

Offset	Register	Reset value
0x00	XOSC Control Register (XOSC_CTL)	0x8030_8000

25.5.3 16 MHz internal RC oscillator (IRCOSC)

The 16 MHz internal RC oscillator default clock is always enabled on reset and can be the clock source for the PLLs. The register interface is for user trimming of the oscillator and dividing output from 1 to 32.

25.5.3.1 IRCOSC register interface

A dedicated digital interface for the IRCOSC has a user register for fine tuning the IRCOSC frequency. Other read-only registers contain settings and values for the IRCOSC temperature sensor, voltage regulator and capacitor trimming.

25.5.3.2 IRCOSC reset value

Table 311 shows the default reset value for the IRCOSC registers.

Table 311. IRCOSC register reset values

Offset	Register	Reset value
0x04	Native Trimming Register (NT)	0x0000_0000
0x08	Temperature Trimming Register (TT)	0x0000_0000

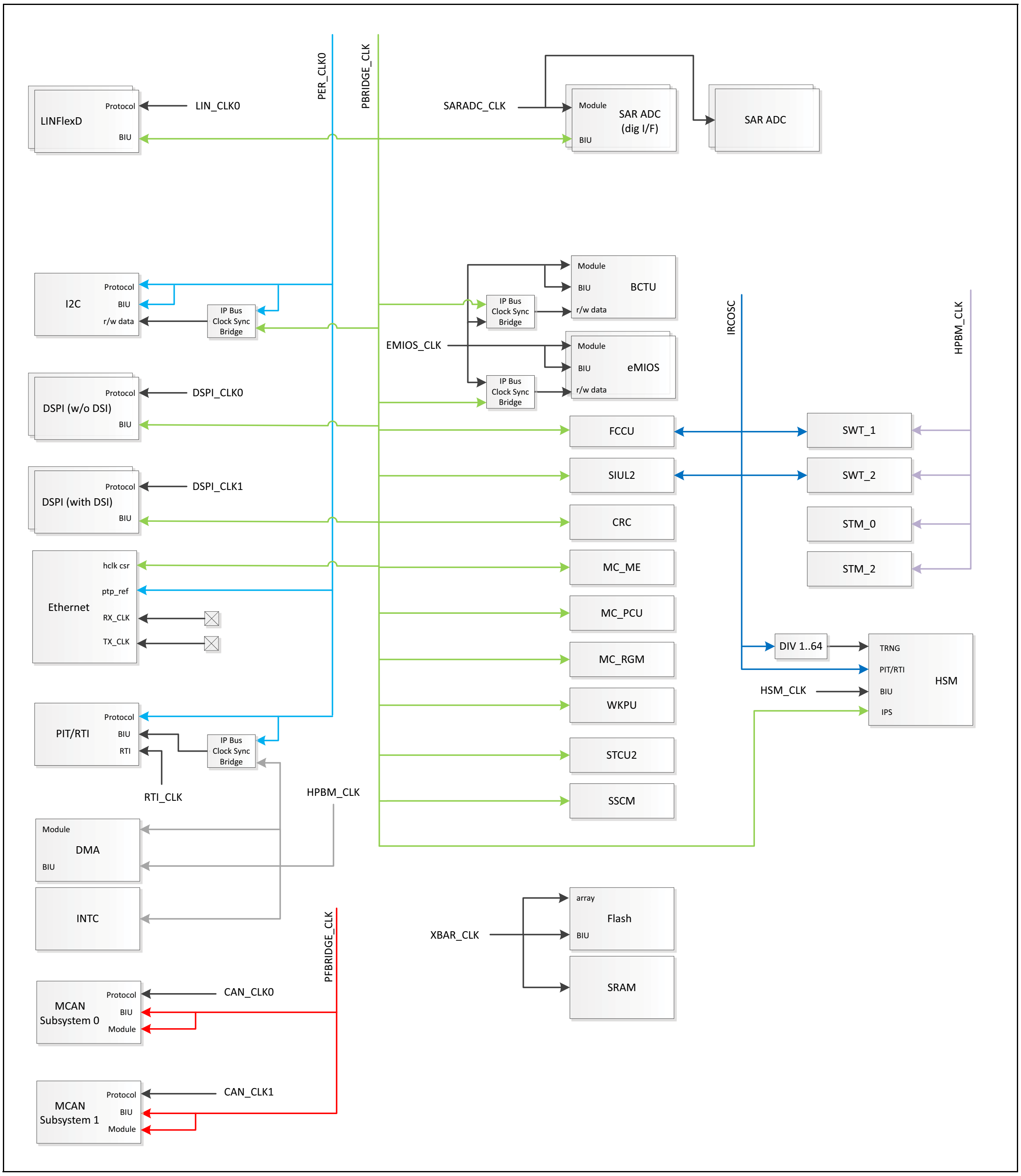
25.5.3.3 IRCOSC register write protection

For IRCOSC register protection, refer to the register protection (REG_PROT) configuration in *Chapter 7: Device configuration*.

25.6 Peripheral clocks

Figure 223 shows the clock distribution to the cores and peripheral modules.

Figure 223. Clock distribution



25.6.1 HSM clock divider

The clock dividers for the HSM core and random number generator (TRNG) are located outside the HSM, but not in the MC_CGM. These dividers are controlled by the HSM. The minimum possible divider after reset is ‘divide by 2’.

The HSM contains an internal PIT/RTI block, which is clocked by the IRCOSC.

The HSM has an internal clock monitor on its input clock, so there is no need for a Clock Monitoring Unit (CMU) for the HSM_CLK in *Figure 220*.

25.6.2 LFAST clocking

The LFAST module is used to support the high speed debug in conjunction with the JTAGM module.

A single LFAST PLL that requires a 10–20 MHz reference supports high speed operation for the LFAST module. For low speed LFAST operation, the reference clock is used directly by the LFAST modules.

The LFAST PLL reference clock source may be selected from multiple sources (AUX Clock selector 1):

- PLL0: PHI output
- external oscillator (XOSC)
- input from the external LFAST device via the DRCLK_IO package pin

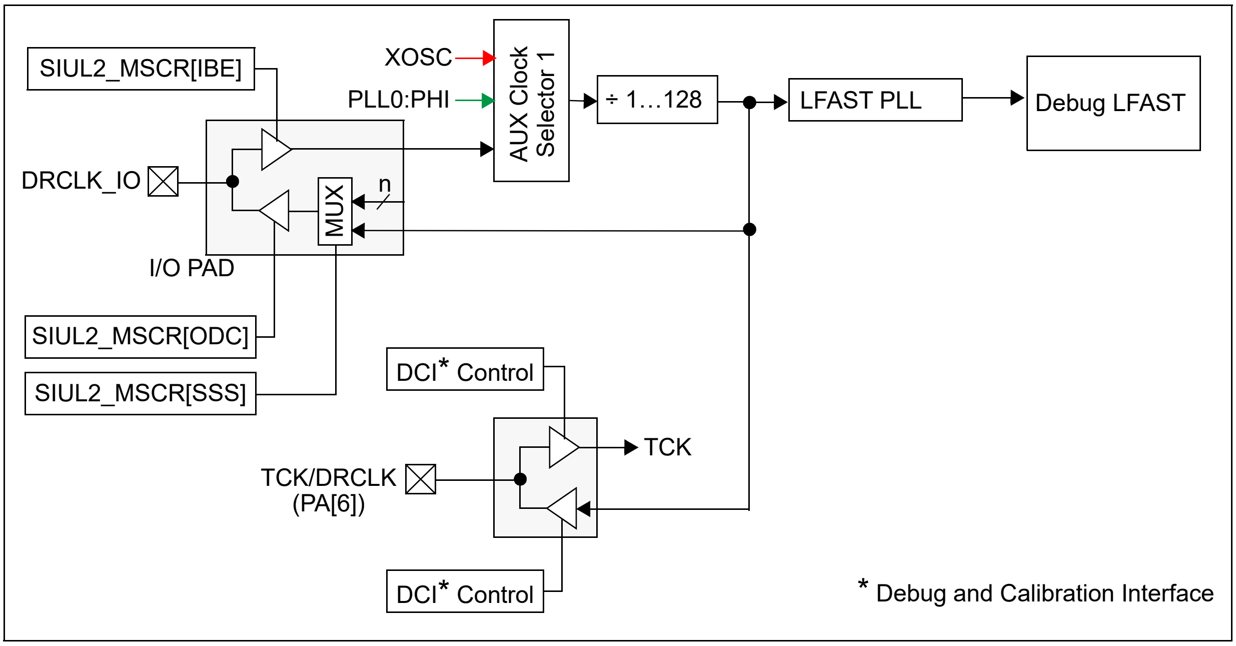
Note that when this reference clock is generated internally, it can be routed as an output onto the DRCLK_IO pin.

In addition, the clock used for the LFAST can also be routed as an output to the JTAG TCK package pin.

The debug LFAST module only supports the operation mode where it outputs the reference clock to a connected LFAST capable tool, as the DRCLK pin only supports output of the DRCLK reference.

Figure 224 shows the clock routing for the various LFAST modules on the device, including connections to the DRCLK_IO and DRCLK pins.

Figure 224. Device LFAST clocking

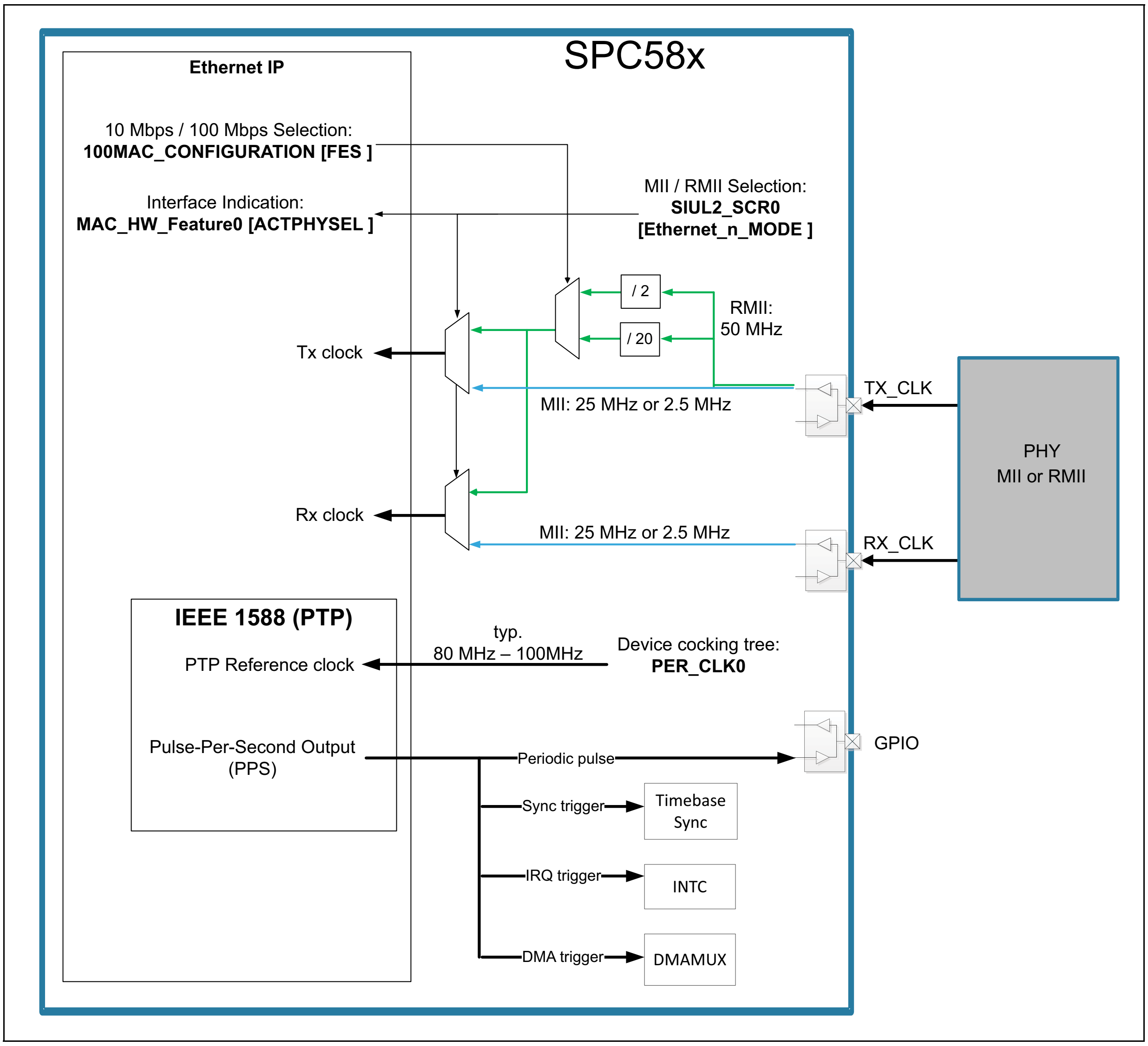


25.6.3 Ethernet clocking

The Ethernet module requires external protocol clock sources for MII and RMII interface modes. The MII clock is 25 MHz in 100 Mbps mode and 2.5 MHz in 10 Mbps mode. The RMII clock is 50 MHz in both modes.

The connections for the Ethernet clocks are as described below.

Figure 225. Ethernet clocking



The RX_CLK pin is an input used in MII mode only and ignored in RMII mode. The TX_CLK pin is an input used in MII mode and RMII mode.

25.6.3.1 Reference Clock for PTP Timestamp Logic

The Ethernet Module supports the Precision Time Protocol (PTP) as defined in IEEE 1588. The PTP Reference clock is the clock at which the PTP system time gets updated. It is connected to PER_CLK0 clock tree provided by the system clock generation module.

Based on the PTP a configurable Pulse-Per-Second signal can be generated as trigger source for several other modules on the device or as periodic pulse as alternate output on a GPIO (refer to *Figure 220*).

25.6.4 M_CAN clocking

25.6.4.1 M_CAN subsystem 0

The CAN_CLK0 is configured in the CGM via the following registers:

- CGM_AC8_SC selects the XOSC or PLL0:PHI as the clock source.
- CGM_AC8_DC0 reduces the CAN_CLK by configuring the dividing factor from 1 to 64.

The CAN_0 clock passes through the Pseudo Random Clock Divider (PRCD) which:

- Divides CAN_CLK by 16.
- Adds approximately 40% of random jitter to both edges of the clock.

PLL0.PHI is the source whenever the PRCD is enabled. PRCD is passive in normal operation and has no impact on the clock frequency.

25.6.4.2 M_CAN subsystem 1

The CAN_CLK1 is configured in the CGM via the following registers:

- CGM_AC11_SC selects the XOSC or PLL0:PHI as the clock source.
- CGM_AC11_DC0 reduces the CAN_CLK by configuring the dividing factor from 1 to 64.

The CAN_1 clock passes through the Pseudo Random Clock Divider (PRCD) which:

- Divides CAN_CLK by 16.
- Adds approximately 40% of random jitter to both edges of the clock.

PLL0.PHI is the source whenever the PRCD is enabled. PRCD is passive in normal operation and has no impact on the clock frequency.

25.6.5 System Clock

The system clock SYSCLK0 has an independent source clock selected from:

- PLL0_PHI and PLL1_PHI.
- Internal oscillator (IRCOSC)
- External oscillator (XOSC)
- RTC clock

The register CGM_AC6_SC configures the clocks for SYSCLK0. Refer to *Figure 220*.

25.7 Clock monitoring

For all safety relevant clocks the microcontroller uses clock monitoring units (CMUs) to detect a missing clock or incorrect frequency.

Each CMU is programmed independently (refer to *Figure 220* for CMU distribution) and uses the IRCOSC or XOSC as the clock monitor reference.

Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

25.7.1 CMU configuration

This section explains the CMU configuration.

Figure 226 shows the block diagram for CMU0 on the SPC584Bx.

Figure 226. CMU0 block diagram

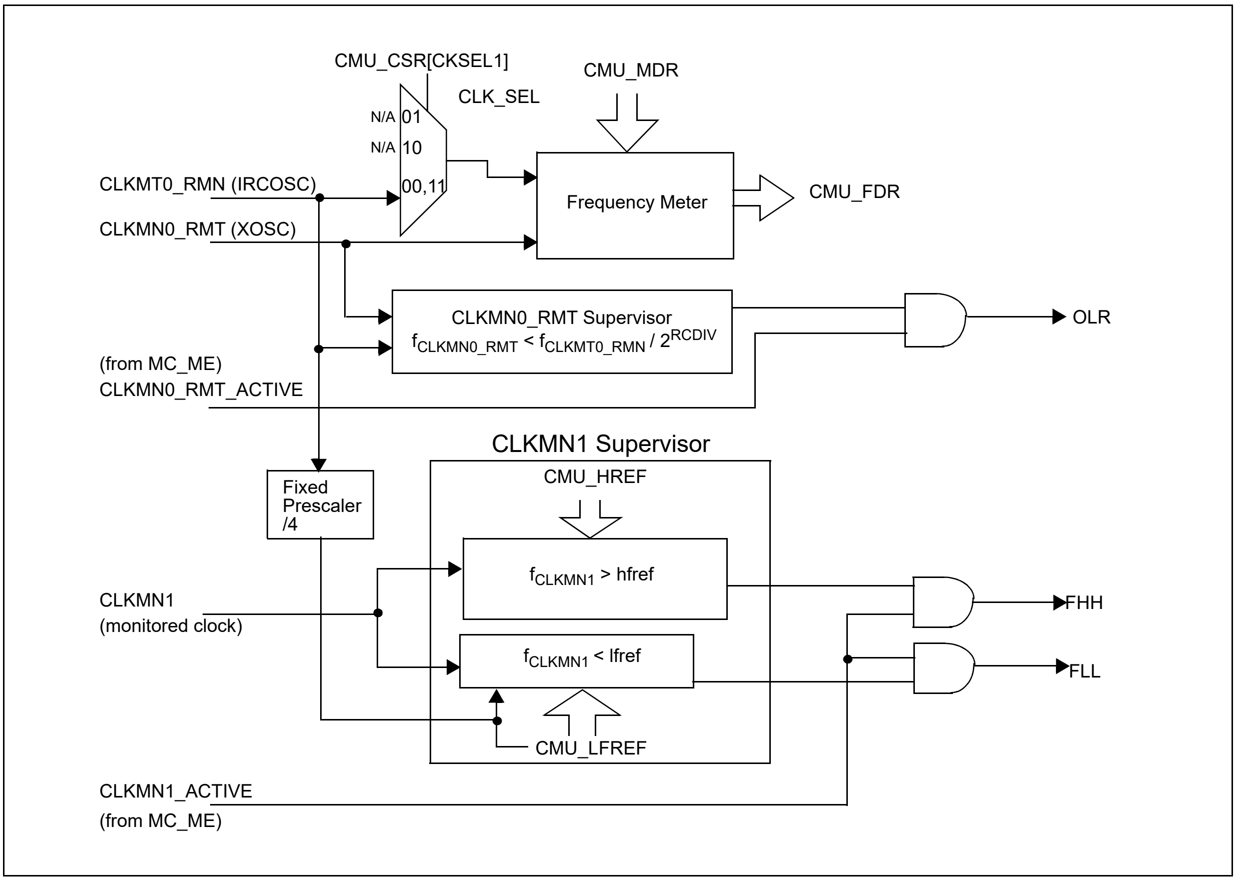
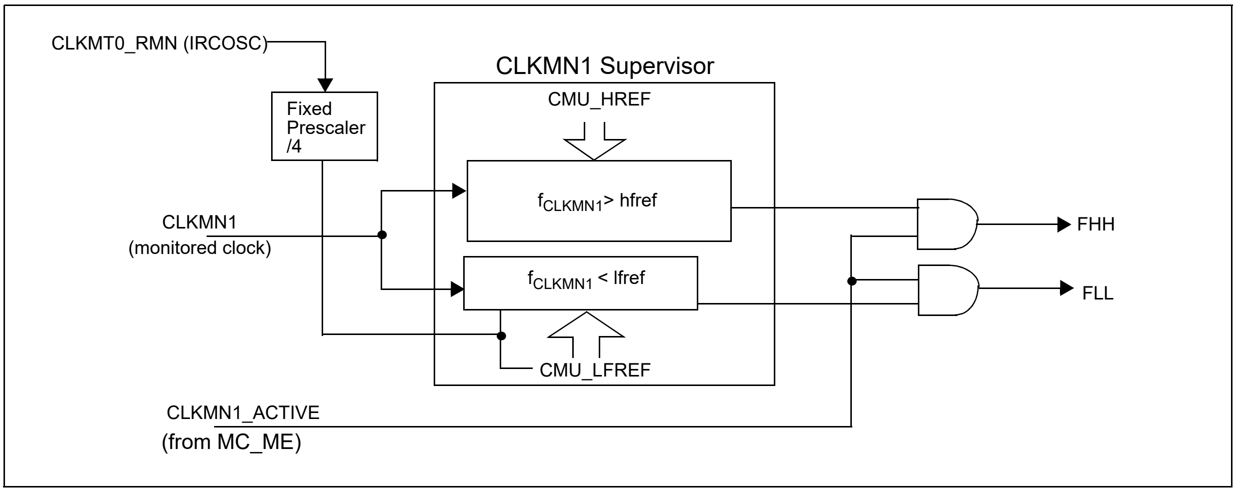


Figure 227 shows the block diagram for the other CMUs on the SPC584Bx.

Figure 227. Other CMUs block diagram



25.7.1.1 Clock input sources

Table 312 shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRCOSC is the reference clock for all clock monitors. Only CMU0 implements the XOSC monitor. CMU0 uses the IRCOSC clock to measure if the XOSC is too low. CMU0 can also be used to calibrate the IRCOSC frequency using the XOSC. All other CMUs are configured identically.

Table 312. Clock input sources

Clock module	Monitored clock	CMU group
CMU_0_PLL0_XOSC_IRCOSC	PLL0:PHI, XOSC, IRCOSC	—
CMU_1_CORE_XBAR	XBAR_CLK	CMU_Platform
CMU_2_HPBM	HPBM_CLK	CMU_Platform
CMU_3_PBRIDGE	PBRIDGE_CLK	CMU_Platform
CMU_6_SARADC	SARADC_CLK	CMU_other
CMU_11_FBRIDGE	FBRIDGE_CLK	CMU_Platform
CMU_12_EMIOS	EMIOS_CLK, BCTU_CLK	CMU_other
CMU_14_PFBRIDGE	PFBRIDGE_CLK	CMU_Platform

25.7.1.2 CMU registers and field availability

Table 313 specifies which registers and fields are available for a given CMU.

Table 313. CMU register availability

Address offset	Register	CMU	Note
0x0000	CMU_CSR	All	CMU_CSR[SFM] and CMU_CSR[CKSEL1] in CMU0 only.
0x0004	CMU_FDR	CMU0	—
0x0008	CMU_HFREFR	All	—
0x000C	CMU_LFREFR	All	—
0x0010	CMU_ISR	All	CMU_ISR[OLRI] in CMU0 only.
0x0014	Reserved	—	—
0x0018	CMU_MDR	CMU0	—

25.7.1.3 CMU register write protection

The CMU registers defined in *Table 314* have write protection with:

- Soft locking: can be unlocked by software after being previously locked.
- Hard locking: can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the CMU register write protection (refer to *Section 7.10.4: Register protection (REG_PROT) configuration* in *Chapter 7: Device configuration* for details).

Table 314. CMU register write protection

Offset	Register[1]
0x0000	CMU Control Status Register (CMU_CSR)
0x0004	CMU Frequency Display Register (CMU_FDR)
0x0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)
0x000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)
0x0010	CMU Interrupt Status Register (CMU_ISR)
0x0014	Reserved
0x0018	CMU Measurement Duration Register (CMU_MDR)

1. Refer to Register Protection chapter for bit field details.

25.7.2 PLL0 monitor

Software can program an upper and lower limit on the expected PLL0 PHI output clock frequency. If the monitor is enabled and the measured frequency is above or below the limits, a flag bit is set and an interrupt, if enabled, is generated. The default condition of the clock monitor is disabled.

25.7.3 External oscillator (XOSC) monitor

The XOSC frequency is compared to a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set and an interrupt, if enabled, is generated.

25.7.4 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

25.7.5 System clock monitors

A CMU is assigned to monitor the frequency of the peripheral bridges, peripheral and ADC clocks (refer to *Figure 220*).

25.8 Loss of system clock behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and provides several options for reaction to a loss of clock in the application. *Figure 228* gives a high-level view of the loss of clock logic.

25.8.1 Loss of PLL/XOSC clock

As shown in *Figure 228*, each loss of lock signal from each PLL and the XOSC failure signals are monitored by the FCCU.

When a clock failure occurs, the FCCU can be programmed to generate:

- A short reset sequence: the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states and the system clock is switched to the IRCOSC.
- A long reset sequence: the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states and the system clock is switched to the IRCOSC.

There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).

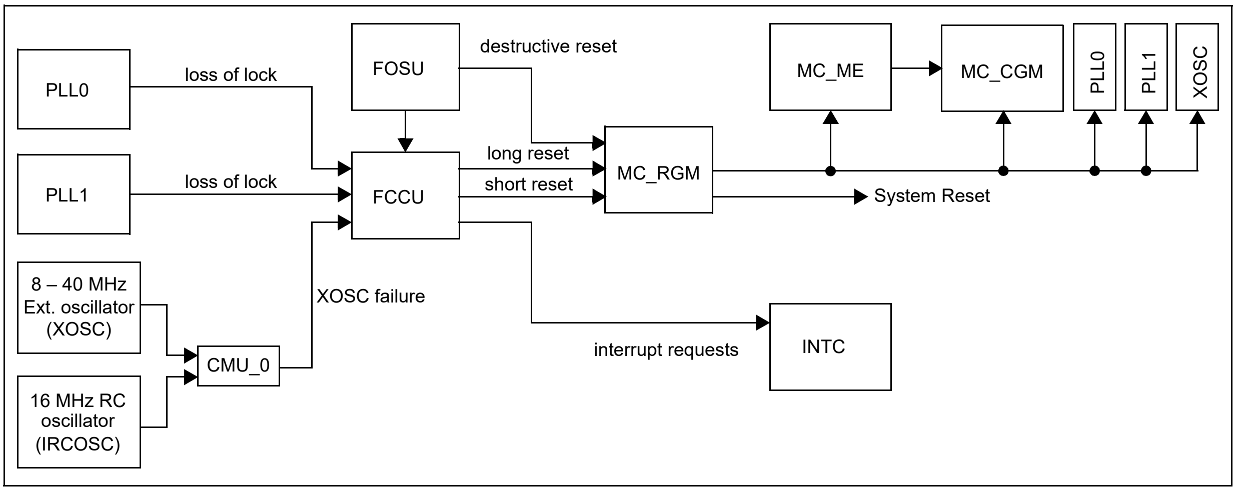
Note: In case of failures related to clocks, device functionality is no more granted till next reset. User should program properly all the safety mechanisms available in oder to reach a device safe-state.

25.8.2 Loss of IRCOSC clock

There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).

The frequency of the IRCOSC clock is monitored by the frequency meter in CMU_0. There is no automated trigger of an FCCU error condition if the IRCOSC fails. Since the IRCOSC is the boot clock, a failure is a catastrophic failure.

Figure 228. Loss of clock logic



25.9 Progressive clock switching

The Progressive Clock Switching (PCS) blocks are shown in the system-level clock diagram in *Figure 220*. They are used for ramping the system. Refer to *Section 28.4.1.2: Progressive system clock switching* from *Chapter 28: Clock Generation Module (MC_CGM)* for more information on the PCS.

26 Dual PLL digital interface (PLLDIG)

26.1 Introduction

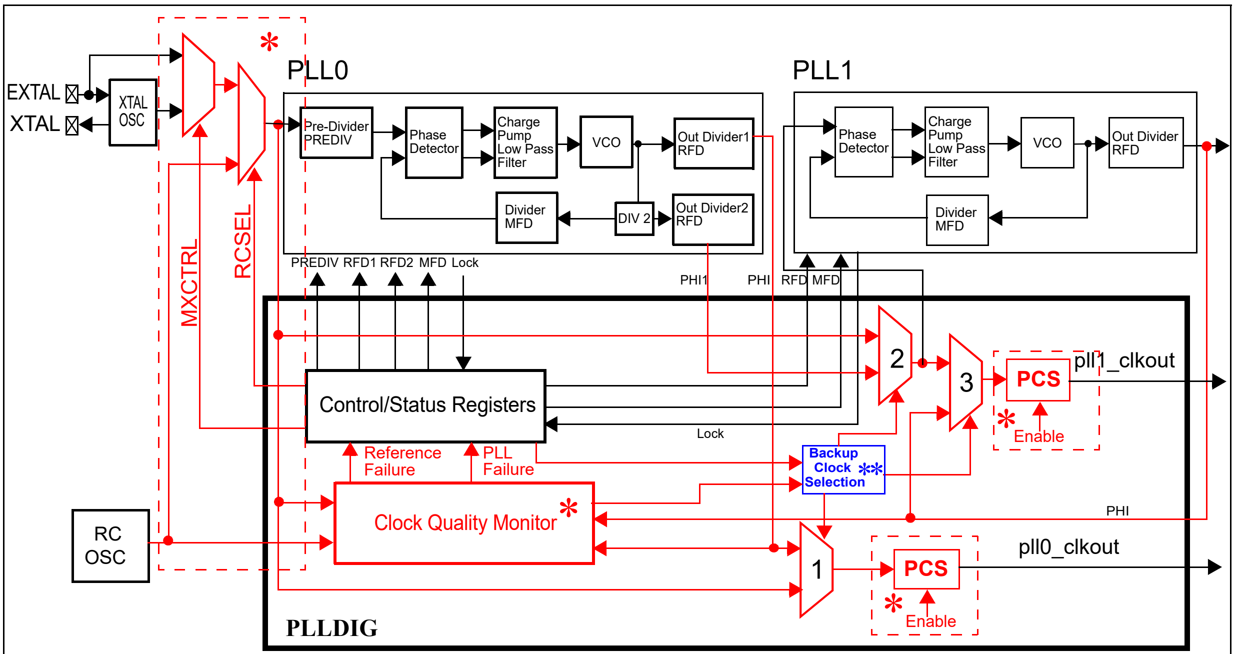
The MCU provides a user interface and control over the dual PLL system composed of PLL0 and PLL1 blocks (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1 as shown in *Chapter 25: Clocking*. The key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by PLL1 PHI frequency modulated output.

Note: For chip-specific implementation details, refer to Chapter 25: Clocking.

26.2 Block diagram

Figure 229 shows the block diagram of the PLL Digital Interface.

Figure 229. Dual PLL digital interface block diagram



* Features in red are not included in all configurations (refer to the Clocking chapter).
** “Backup Clock Selection” block in blue is not implemented.

26.3 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for reducing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Programmable frequency modulation on PLL1.
- The frequency range after the prescaler is required to be 8-20Mhz.
- Lock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
  - User-selectable ability to generate an interrupt request upon loss of lock.
  - The loss of lock indication is sent to the FCCU via the system glue logic.

Note: Please refer to the device datasheet for information on the input and output clock frequency range.

26.4 Modes of operation

The operating mode of the PLLs is determined by the value of PLL*n*CR[CLKCFG]. The mode of operation for the PLL is defined below:

Normal mode with reference, and either PLL0 or both PLLs enabled.

Note: Mode changes are controlled by configuring the Mode Entry mode configuration registers (<ME_mode>_MC).

Normal mode is defined as the mode where the clocks are driven by the PLL. When using a crystal for the clock reference, reset should remain asserted until the oscillator has stabilized.

When PLL0 is powered down and working in bypass mode, PLL1 should be bypassed and powered down. PLL1 should be configured to work in normal mode only after PLL0 has achieved lock (for example, bypass with PLL0 running or PLL0 in normal mode, PLL*n*SR[LOCK] = 1).

26.4.1 Normal mode with reference, PLL0 or both PLLs enabled

In the normal mode, PLL0 receives an input clock from the reference and the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0_PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, output divider (Reduced Frequency Divider) and whether the PLL1 is modulated or not can be programmed by writing to the PLL1FM register, *Section 26.5.2.7: PLL1 frequency modulation register (PLL1FM)*.

Note: While powering down the PLLs, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Note: Only when the PLL0 achieves lock should PLL1 be configured to work in either mode by programming the appropriate <ME_mode>_MC register (see Section 26.7: Initialization information).

26.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs.

26.5.1 Memory map

Table 315 shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

Table 315. Dual PLL digital interface memory map

Offset (hex)	Register	Section
0x0000	PLL0 Control Register (PLL0CR)	*Section 26.5.2.1*
0x0004	PLL0 Status Register (PLL0SR)	*Section 26.5.2.2*
0x0008	PLL0 Divider Register (PLL0DV)	*Section 26.5.2.3*
0x000C–0x001F	Reserved	
0x0020	PLL1 Control Register (PLL1CR)	*Section 26.5.2.4*
0x0024	PLL1 Status Register (PLL1SR)	*Section 26.5.2.5*
0x0028	PLL1 Divider Register (PLL1DV)	*Section 26.5.2.6*
0x002C	PLL1 Frequency Modulation Register (PLL1FM)	*Section 26.5.2.7*
0x0030	PLL1 Fractional Divide Register (PLL1FD)	*Section 26.5.2.8*
0x0034–0x003F	Reserved	

26.5.2 Register descriptions

Some of the register reset values are specifically configured for each unique device by external configuration signals or parameters.

26.5.2.1 PLL0 control register (PLL0CR)

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CLKCFG(3)		EXPDIE	0(1)	0(1)	0(1)	LOLIE(2)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. Refer to the Clocking chapter for details on loss of lock management.
3. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate <ME_mode>_MC register.

Figure 230. PLL0 control register (PLL0CR)

Table 316. PLL0CR field descriptions

Field	Description
22:23 CLKCFG	Clock configuration This field indicates the operational state of PLL. 00 PLL off. 01 Reserved 10 Reserved. 11 Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b. **Note:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.
24 EXPDIE	External power down cycle complete indication interrupt enable This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion and enable the FCCU failure "Loss of lock flag PLL0"
28 LOLIE	Loss of lock interrupt enable The LOLIE bit enables a loss of lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss of lock. Interrupt not requested 1 Enable interrupt request upon loss of lock

26.5.2.2 PLL0 status register (PLL0SR)

Offset: 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Figure 231. PLL0 status register (PLL0SR)

Table 317. PLL0SR field descriptions

Field	Description
24 EXTPDF	External power down cycle complete indication interrupt flag This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
28 LOLF	Loss of lock flag This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested 1 Loss of lock detected. Interrupt service requested
29 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked 1 PLL is locked

26.5.2.3 PLL0 divider register (PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider.
The values of PREDIV and MFD should not be changed when the PLL is on and locked (working in Normal mode). If these fields are changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled.
The reduced frequency divider fields (RFDPHI, RFDPHI1) can be modified at anytime, but the changes only become effective after PLL0 is disabled, then re-enabled.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RFDPHI1				0	0	0	0	0	RFDPHI					
W																
Reset	0	_(1)	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PREDIV			0	0	0	0	0	MFD						
W																
Reset	0	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. Refer to the Clocking chapter for default reset value information.

Figure 232. PLL0 divider register (PLL0DV)

Table 318. PLL0DV field descriptions

Field	Description
1:4 RFDPHI1	PHI1 reduced frequency divider This 4-bit field determines the VCO/2 clock post divider for driving the PHI1 output clock. 00xx Reserved 0100 Divide by 4 0101 Divide by 5 ... 1110 Divide by 14 1111 Divide by 15
10:15 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 00 Reserved 01 Divide by 1 02 Divide by 2 03 Divide by 3 ... 3E Divide by 62 3F Divide by 63

Table 318. PLL0DV field descriptions (continued)

Field	Description
17:19 PREDIV	Input clock pre-divider This 3-bit field controls the value of the divider on the input clock. The output of the pre-divider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 ... 110 Divide by 6 111 Divide by 7
25:31 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 00 Reserved ... 07 Reserved 08 Divide by 8 ... 7E Divide by 126 7F Divide by 127

26.5.2.4 PLL1 control register (PLL1CR)

Offset: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CLKCFG(3)		EXPDIE	0(1)	0(1)	0(1)	LOLIE(2)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. Refer to the Clocking chapter for details on loss of lock management.
3. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate <ME_mode>_MC register.

Figure 233. PLL1 control register (PLL1CR)

Table 319. PLL1CR field descriptions

Field	Description
22:23 CLKCFG	Clock configuration This field indicates the operational state of PLL. 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b. **Note:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.
24 EXPDIE	External power down cycle complete indication interrupt enable This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
28 LOLIE	Loss of lock interrupt enable The LOLIE bit enables a loss of lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss of lock. Interrupt not requested 1 Enable interrupt request upon loss of lock and enable the FCCU failure"Loss of clock flag PLL1"

26.5.2.5 PLL1 status register (PLL1SR)

Offset: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Figure 234. PLL1 status register (PLL1SR)

Table 320. PLL1SR field descriptions

Field	Description
24 EXTPDF	External power down cycle complete indication interrupt flag This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
28 LOLF	Loss of lock flag This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested 1 Loss of lock detected. Interrupt service requested
29 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked 1 PLL is locked

26.5.2.6 PLL1 divider register (PLL1DV)

The PLL1DV register provides the output clock reduced frequency dividers and loop divider values.
The value of MFD should not be changed when the PLL is on and locked (working in Normal mode). If the field is changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled.
The reduced frequency divider field (RFDPHI) can be modified at anytime, but the changes only become effective after PLL1 is disabled, then re-enabled.

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	RFDPHI(1)					
W																
Reset	0	0	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	0	0	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	MFD(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)

Figure 235. PLL1 divider register (PLL1DV)

1. Refer to the Clocking chapter for default reset value information.

Table 321. PLL1DV field descriptions

Field	Description
10:15 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 0x00 Reserved 0x01 Divide by 1 0x02 Divide by 2 ... 0x3E Divide by 62 0x3F Divide by 63
25:31 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 0x10 Divide by 16 0x11 Divide by 17 .... 0x21 Divide by 33 0x22 Divide by 34 All other settings are reserved

26.5.2.7 PLL1 frequency modulation register (PLL1FM)

The PLL1FM register enables frequency modulation on PLL1 and provides controls for the FM spread spectrum, modulation depth and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down).
Changing the values of PLL1FM[MODEN] and PLL1FM[MODSEL] fields once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (if PLL1CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality.
The modulation period (PLL1FM[MODPRD]) and increment step fields (PLL1FM[INCSTP]) can be changed without losing lock, however, the output clock from PLL may have an incorrect frequency for a few cycles after either of these updates (as defined in the datasheet).

Offset: 0x002C Access: User read/write

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MODEN	MODSEL	MODPRD												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	INCSTP														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 236. PLL1 frequency modulation register (PLL1FM)

Table 322. PLL1FM field descriptions

Field	Description
1 MODEN	Modulation enable This bit enables the frequency modulation. 0 Frequency modulation disabled 1 Frequency modulation enabled
2 MODSEL	Modulation selection This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Modulation centered around nominal frequency 1 Modulation spread below nominal frequency
3:15 MODPRD	Modulation period This 13-bit field is the binary equivalent of the modulation period variable derived from the formula: $$MODPRD = round(\frac{f_{ref}}{4 \times f_{mod}})$$ where f_{ref} represents the frequency of the feedback divider and f_{mod} represents the modulation frequency (refer to equation in *Section 26.6.3*).
17:31 INCSTP	Increment step This 15-bit field is the binary equivalent of the INCSTP variable derived from the formula: $$INCSTEP = round(\frac{(2^{15} - 1) \times MD \times PLL1CR[MFD]}{100 \times 5 \times MODPRD})$$ where *md* represents the peak modulation depth in percentage (+/- *MD* for centered modulation, -2**MD* for modulation below nominal frequency), and MFD represents the nominal value of the feedback loop divider (refer to equation in *Section 26.6.3*).

26.5.2.8 PLL1 fractional divider register (PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (if PLL1CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality.
The dither disable and fractional divider fields can be changed without losing lock, however, the output clock from PLL can have an incorrect frequency for a few cycles after either of these updates .

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FDEN	0	0	0	0	0	0	0	0	0	0	0	0	DTHDIS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	FRCDIV											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 237. PLL1 fractional divider register (PLL1FD)

Table 323. PLL1FD field descriptions

Field	Description
1 FDEN	Fractional divide enable This bit enables the fractional divider in the loop divider for PLL1. 0 Fractional divide disabled 1 Fractional divide enabled
14:15 DTHDIS	Dither disable Bit 14 and bit 15 control noise of SSCG and Fractional controller. 00 Increase PLL multiplication factor by 1/2^13 and has maximum noise control of SSCG and Fractional controller. 01 Increase PLL multiplication factor by 1/2^13 and has some noise control of SSCG and Fractional controller. 10 No influence on PLL multiplication factor and has some noise control of SSCG and Fractional controller. 11 No influence on PLL multiplication factor and no noise control of SSCG and Fractional controller.
20:31 FRCDIV	Fractional divide input When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the mean VCO clock frequency is given by: $\text{VCO Frequency} = \text{InputClock} \times (\text{PLL1DV[MFD]} + (\text{FRCDIV} \div 2^{12}))$ where FRCDIV is the decimal equivalent of the FRCDIV bits (refer to *Section 26.6.2*). **Note:** Depending on DTHDIS value the multiplication factor can change as described in DTHDIS field description.

26.6 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

26.6.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for each PLL are discussed in detail in the *SPC584Bx datasheet*.

26.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV and PLL1FD registers, and calculated according to the following equations:

Equation 4

$$f_{pll0\_phi} = f_{pll0\_ref} \times \frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]}$$

Equation 5

$$f_{pll0\_phi1} = f_{pll0\_ref} \times \frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]}$$

Equation 6

$$f_{pll1\_phi} = f_{pll1\_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Note: Depending on DTHDIS value the multiplication factor of *Equation 6* can change as described in PLL1DV[DTHDIS] field description (refer to *Table 323*).

The relationship between the VCO frequency f_{VCO} and the output frequency of the PLLs is determined by the configuration of the PLL0DV, PLL1DV and PLL1FD registers, according to the following equations:

Equation 7

$$f_{pll0\_VCO} = \frac{f_{pll0\_ref} \times PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]}$$

Equation 8

$$f_{pll1\_VCO} = f_{pll1\_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} \right)$$

Note: *Depending on DTHDIS value the multiplication factor of Equation 8 can change as described in PLL1DV[DTHDIS] field description (refer to Table 323).*

Note: *f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1.*

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification for PLL0 or PLL1 (refer to the device datasheet). Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of PLL*n*DV are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLL*n*CR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

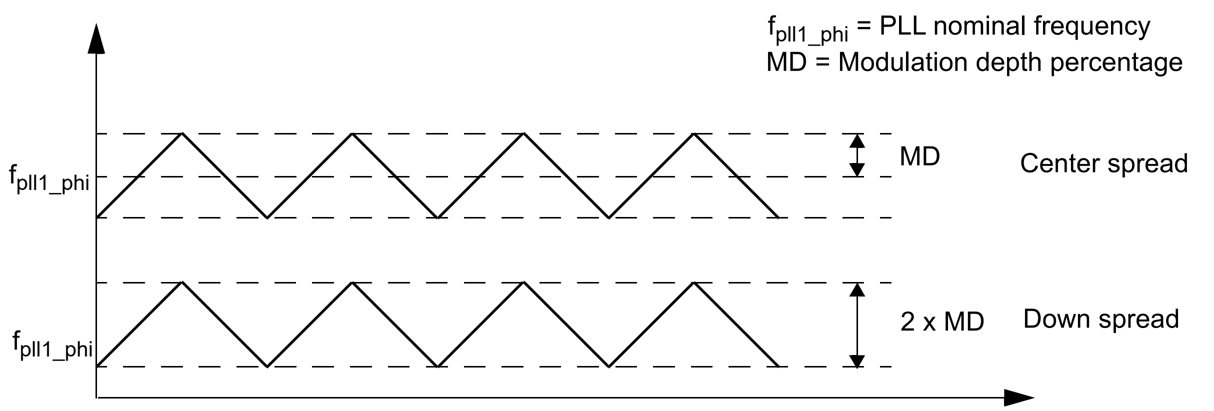
Note: The PLL must be powered down and powered up by configuring <ME_mode>_MC before PLL0CR[PREDIV], PLL0CR[MFD], PLL1CR[MFD] or the input reference clock values are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in *Section 26.7*.

26.6.3 Frequency modulation

Frequency modulation uses a triangular profile as shown in *Figure 238*. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

Figure 238. Triangular frequency modulation



Note: The device maximum operating frequency includes the frequency modulation. If center modulation is selected, the fsys must be below fmax by MD percentage such that fmax = fsys(1 + MD%). Refer to the Analog PLL chapter for more details.

The following equations define how to calculate PLL1FM[MODPRD] (*Equation 9*) and PLL1FM[INCSTP] (*Equation 10*) based on the frequency of the feedback divider (f_{pll1_ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD):

Equation 9

$$PLL1FM[MODPRD] = round\left(\frac{f_{pll1\_ref}}{4 \times f_{mod}}\right)$$

Equation 10

$$PLL1FM[INCSTP] = round\left(\frac{(2^{15} - 1) \times MD \times PLL1DV[MFD]}{100 \times 5 \times PLL1FM[MODPRD]}\right)$$

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

Equation 11

$$(PLL1FM[MODPRD] \times PLL1FM[INCSTP]) < 2^{15}$$

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is given by the following formula:

Equation 12

$$ModulationDepth = round\left(\frac{PLL1FM[MODPRD] \times PLL1FM[INCSTP] \times 100 \times 5}{(2^{15} - 1) \times PLL1DV[MFD]}\right)$$

As an example, suppose the following configuration:

- Input frequency: 40 MHz
- Loop divider (PLL1DV[MFD]): 20
- Input divider: 1
- Output divider RFD: 5
- VCO frequency: 40 MHz × 20 = 800 MHz
- PLL output frequency: 800 MHz / 2*RFD = 800 MHz / 10 = 80 MHz
- Center spread (PLL1FM[MODSEL] = 0)
- Modulation frequency: 200 kHz
- Modulation depth: ± 1.9% (3.8% peak-to-peak)
- PLL1FM[MODPRD] = Round[(40 × 10^6) / (4 × 200 × 10^3)] = Round[50] = 50
- PLL1FM[INCSTP] = Round[((2^{15} – 1) × 1.9× 20) / (100 × 5 × 50)] = Round[49.8] = 50
- PLL1FM[MODPRD] × PLL1FM[INCSTP] = 50 × 50 = 2500 (which is less than 2^{15})
- MD (quantized) = ((50 × 50 × 100 × 5) / ((2^{15} – 1) × 20) = 1.90740%

In this example, the modulation error is 0.00740%.

The FM parameters do not get reset when PLL0 looses its lock. These parameters get reset on destructive reset. Whenever PLL0 gets relocked FM modulation starts with the previous values. The sequence for reprogramming the FM is:

1. Power down PLL1[h]
2. Program the PLL1FM while PLL1 is powered down
3. Power up the PLL1[i]

h. By writing <ME_mode>_MC[PLL1ON] = 0b.

i. By writing <ME_mode>_MC[PLL1ON] = 1b.

26.7 Initialization information

Coming out of reset the PLLs are disabled per the DRUN mode configuration register (<*ME*>_DRUN_MC). The recommended procedure[j] to program the PLLs and enter normal mode is:

1. Configure PLL0 and related modules.

   Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].

   a) With PLL0 disabled, program PLL0 clock source in <CGM_ACn>_SC[SELCTL]. Default source is 16 MHz IRCOSC. Write <CGM_ACn>_SC[SELCTL] = 1 to select XOSC as source.

   b) Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].

   c) If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the Mode Entry module).

   d) If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.

2. Turn on XOSC and PLL0.

   a) Configure a mode configuration for turning on PLL0 and XOSC. In a mode configuration register (for example, <ME>_DRUN_MC) set <ME_mode>_MC[XOSCON] = 1 and <ME_mode>_MC[PLL0ON] = 1. If desired, also set <ME_mode>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0 (PLL0 PHI output) as the sysclk.

   b) Enter that mode by two writes to <ME>_CTRL register to enter that mode. This is required even if re-entering the same mode.

3. Wait for the mode transition to complete.

   a) Wait for the mode transition to complete by polling <ME>_GS[S_MTRANS] or enabling an interrupt for flag <ME>_IS[I_MTC]. A timer, even if it is the watchdog,

j. The steps from 1 to 6 describe a recommended way to initialize the PLLs. But the mode_transition from DRUN to RUNx with the PLLs ON (with PLL1 as sys_clk and PLL0_PHI1 as source of PLL1 and all other clk sources enabled) can be done with a single mode transition. The only software requirement is to ensure that the reference clocks are also configured ON when the derived clock is configured ON in the target mode.

should be used to make sure the mode transition completes. Mode transition will NOT complete until:

- XOSC counter expires (if the new mode configuration changes <ME_mode>_MC[XOSCON] to 1), and

- PLL is locked (if the new mode configuration changes <ME_mode>_MC[PLL0ON] = 1).

   b) Confirm the desired target mode was entered by checking the status of <ME>_GS[S_CURRENT_MODE].

4. Configure PLL1 and related modules
   a) With PLL1 disabled, program PLL1 clock source in <CGM_ACn>_SC[SELCTL]. Default is <CGM_ACn>_SC[SELCTL] = which selects XOSC.
   b) Program PLL1DV[MFD] and PLL1DV[RFDPHI].
   c) If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.
5. Turn on PLL1
   a) Configure a mode configuration for turning on PLL1, and keeping XOSC and PLL0 on. In a mode configuration register (for example, <ME>_DRUN_MC), initialize <ME_mode>_MC[XOSCON] = 1, <ME_mode>_MC[PLL0ON] = 1 and <ME_mode>_MC[PLL1ON] = 1. If desired, also set <ME_mode>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.
   b) Enter that mode by two writes to <ME>_CTRL register to enter that mode. This is required even if re-entering the same mode.
6. Wait for the mode transition to complete.
   a) Wait for the mode transition to complete by polling <ME>_GS[S_TRANS] or enabling an interrupt for flag <ME>_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

      XOSC counter expires (if the new mode configuration changes <ME_mode>_MC[XOSCON] = 1), and PLL1 is locked (if the new mode configuration changes <ME_mode>_MC[PLL1ON] = 1).
   b) Confirm the desired target mode was entered by checking the status of <ME>_GS[S_CURRENT_MODE].

Note: Refer to the Mode Entry module chapter for more details.

27 Clock Monitor Unit (CMU)

27.1 Introduction

The Clock Monitor Unit (CMU), also referred to as clock quality checker or clock fault detector, serves three purposes:

- Measures the frequency of clock sources CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and also detects if the monitored clock frequency leaves an upper or lower frequency boundary

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the division factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate events when the CLKMN1 frequency is less than the CLKMT0_RMN frequency divided by four or if CLKMN1 leaves an upper or lower frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of one clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

Note: Refer to the clocking chapter for the number of instances of the CMU in this device.

27.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as the reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN/n clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to CLKMT0_RMN / 4.
- Event generation for various failures detected inside monitoring unit.

27.2 Block diagram

The block diagram of the CMU module(s) is shown in the clocking chapter of this reference manual.

27.3 Signals

Table 324 describes the signals on the boundary of the CMU (in alphabetical order).

Table 324. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference: Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1: Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference: Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

Note: Refer to the clocking chapter for device specific clock sources of each CMU.

27.4 Memory map and register definition

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in Table 325.

Table 325. CMU memory map

Address offset	Register	Location
0x0000	CMU Control Status Register (CMU_CSR)	Section 27.4.1.1
0x0004	CMU Frequency Display Register (CMU_FDR)	Section 27.4.1.2
0x0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	Section 27.4.1.3
0x000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	Section 27.4.1.4
0x0010	CMU Interrupt Status Register (CMU_ISR)	Section 27.4.1.5
0x0014–0x0017	Reserved	
0x0018	CMU Measurement Duration Register (CMU_MDR)	Section 27.4.1.6

Note: See Clocking chapter for register and field availability details.

27.4.1 Register descriptions

27.4.1.1 CMU Control Status Register (CMU_CSR)

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SFM(1)	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CKSEL1(1)		0	0	0	0	0	RCDIV(1)		CME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 239. CMU Control Status Register (CMU_CSR)

1. Not all CMU blocks will utilize this feature. Refer to clocking chapter for device specific CMU implementation details

Table 326. CMU_CSR field descriptions

Field	Description
8 SFM	Start Frequency Measure The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR register. CMU_MDR[MD] must be written before enabling frequency measurement (CMU_CSR[SFM] = 1). Do not write CMU_MDR[MD] while CMU_CSR[SFM] = 1. Software should only read the value of CMU_FDR[FD] bits once the hardware has cleared the CMU_CSR[SFM] bit. 0 Frequency measurement is completed or not yet started. 1 Frequency measurement is not completed.
22:23 CKSEL1	Frequency measure clock selection bit CKSEL1 selects the clock to be measured by the frequency meter. This only effects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected. 01 Reserved 10 Reserved. 11 CLKMT0_RMN is selected.

Table 326. CMU_CSR field descriptions (continued)

Field	Description
29:30 RCDIV	CLKMT0_RMN division factor These bits specify the CLKMT0_RMN division factor. The output clock frequency is f_{CLKMT0_RMN} divided by the factor 2^{RCDIV}. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. Ensure clock source for CLKMN0_RMT is off before writing CMU_CSR[RCDIV] otherwise the operation can lead to a spurious OLR event. The clock division coding is as follows: 00 CLKMT0_RMN divided by 1 (No division). 01 CLKMT0_RMN divided by 2. 10 CLKMT0_RMN divided by 4. 11 CLKMT0_RMN divided by 8.
31 CME	CLKMN1 monitor enable CMU_HFREF and CMU_LFREF registers must be written before enabling the clock monitoring (CMU_CSR[CME] =1) 0 CLKMN1 monitor is disabled. 1 CLKMN1 monitor is enabled.

27.4.1.2 CMU Frequency Display Register (CMU_FDR)

The FDR is used to determine the measured frequency being monitored by the CMU.

Offset: 0x0004 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	FD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 240. CMU Frequency Display Register (CMU_FDR)

Table 327. CMU_FDR field descriptions

Field	Description
12:31 FD	Measured frequency bits This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: f_{sel} = (f_{CLKMN0_RMT} × CMU_MDR[MD]) / CMU_FDR[FD]

27.4.1.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. *Figure 241* and *Table 328* show the CMU_HFREFR register.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	HFREF											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 241. CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

Table 328. CMU_HFREFR field descriptions

Field	Description
20:31 HFREF	High Frequency reference value These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF / 16) × (f_{CLKMT0_RMN} / 4)

27.4.1.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. *Figure 242* and *Table 329* show the CMU_LFREFR register.

Offset 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	LFREF											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 242. CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

Table 329. CMU_LFREFR field descriptions

Field	Description
20:31 LFREF	Low Frequency reference value These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF / 16) × (f_{CLKMT0_RMN} / 4)

27.4.1.5 CMU Interrupt Status Register (CMU_ISR)

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FHHI	FLLI	OLRI(1)
W													—(2)	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 243. CMU Interrupt Status Register (CMU_ISR)

1. Not all CMU blocks will utilize this feature. See Clocking chapter for device specific CMU implementation details.
2. Field is read only.

Table 330. CMU_ISR field descriptions

Field	Description
29 FHHI	CLKMN1 frequency higher than high reference event status This bit is set by hardware when CLKMN1 frequency becomes higher than HFREF value and CLKMN1 is ‘ON’ as signaled by the MC_ME. It can be cleared by software by writing ‘1’. 0 No FHH event 1 FHH event occurred
30 FLLI	CLKMN1 frequency less than low reference event status This bit is set by hardware when CLKMN1 frequency becomes lower than LFREF value and CLKMN1 is ‘ON’ as signaled by the MC_ME. It can be cleared by software by writing ‘1’. 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than f_{CLKMT0_RMN} / 2^{RCDIV} event status This bit is set by hardware when the f_{CLKMN0_RMT} is less than f_{CLKMT0_RMN} / 2^{RCDIV} frequency and CLKMN0_RMT is ‘ON’ as signaled by the MC_ME. It can be cleared by software by writing ‘1’. 0 No OLR event 1 OLR event occurred

Note: All the flags in CMU_ISR are set asynchronously. This register must be read only after a fault event is triggered by the CMU and mapped on FCCU.

Otherwise, the read access on this register may fetch an incorrect value.

Before entering low-power stop modes, all clock frequency measurements in CMU should be disabled. Failing to do so may result in spurious interrupts.

27.4.1.6 CMU Measurement Duration Register (CMU_MDR)

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 244. CMU Measurement Duration Register (CMU_MDR)

Table 331. CMU_MDR field descriptions

Field	Description
12:31 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1.

27.5 Functional description

This section describes the functionality of the CMU.

27.5.1 Frequency meter

The purpose of frequency meter is to evaluate the deviation from the nominal metered source (such as, CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the Clocking chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR as shown in the following equation:

Equation 13 $\mathbf{f_{sel} = (f_{CLKMN0\_RMT} \times MDR[MD]) / FDR[FD]}$

27.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than frequency of CLKMT0_RMN divided by $2^{CMU\_CSR[RCDIV]}$ and CLKMN0_RMT is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

27.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If fCLKMN1 is greater than the reference value determined by field CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

Note: An example of determining the HFREFActual value is as follows. Assume a fCLKMT0_RMN = 16 MHz with an accuracy of +/-5%. In order to monitor *fCLKMN1 = 200 MHz, the ideal HFREFIdeal = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration (HFREFActual = (HFREFIdeal + 2) × 1.05).*

IIf fCLKMN1 is lower than CMU_LFREFR[LFREF] and CLKMN1 is ‘ON’ as signaled by the MC_ME, then

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note: The low reference value must be programmed considering the following factors:

- *Two cycles of f_{CLKMN1} which is the built-in tolerance of the monitor implementation*
- *Frequency variation % of f_{CLKMT0_RMN} across PVT*

For example, assuming f_{CLKMT0_RMN} is 16 MHz with +/-5% variation, in order to monitor the f_{CLKMN1} (200 MHz), the ideal $LFREF_{Ideal}$ is 800. The adjusted value considering the tolerances will be 758 ($LFREF_{Actual}$ = ($LFREF_{Ideal}$ - 2) x 0.95).

The minimum value that can be programmed in CMU_LFREFR[LFREF] is 3. The minimum frequency that can be monitored using the CLKMN1 supervisor must be >= f_{CLKMT0_RMN} / 16.

28 Clock Generation Module (MC_CGM)

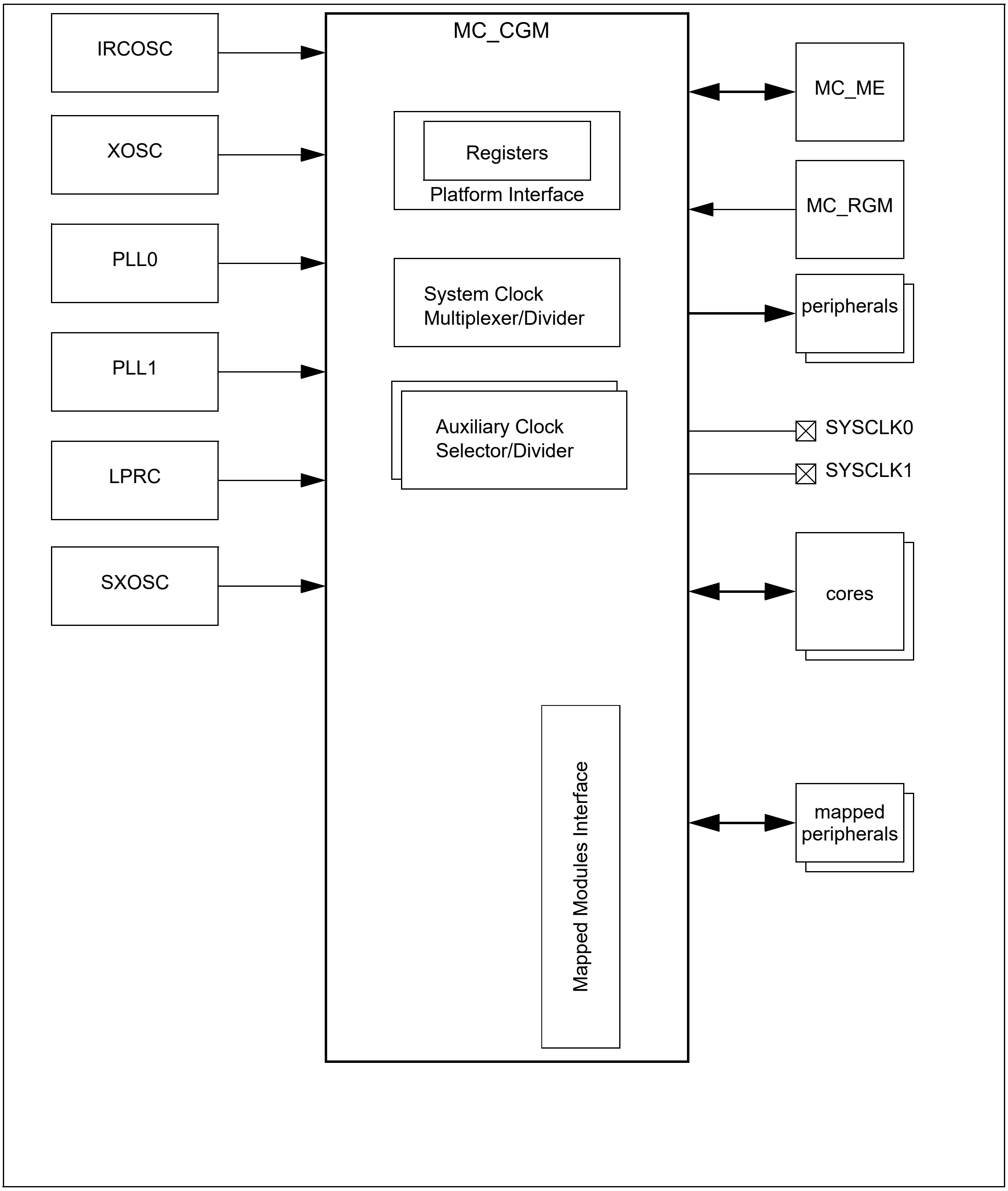
28.1 Introduction

28.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the modules on the chip. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space.

Figure 245 shows the MC_CGM block diagram.

Figure 245. MC_CGM block diagram



28.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control
- performs progressive system clock frequency change depending on MC_ME mode configuration
- contains a set of registers to control clock dividers for divided clock generation
- supports multiple clock sources and maps their address spaces to its memory map
- guarantees glitch-less clock transitions when changing the system clock selection
- supports 8, 16, and 32-bit wide read/write accesses

28.2 External signal description

The MC_CGM delivers output clocks to the SYSCLK0 and SYSCLK1 pins for off-chip use and/or observation.

28.3 Memory map and register definition

Note: Any access to unused registers as well as write accesses to read-only registers will:

- *not change register content*
- *cause a transfer error*

Table 332. MC_CGM memory map

Address offset	Register	Location
0x0040–0x00FF	Reserved	
0x0100	PCS Switch Duration Register (CGM_PCS_SDUR)	Section 28.3.1.1
0x0104	PCS Divider Change Register 1 (CGM_PCS_DIVC1)	Section 28.3.1.2
0x0108	PCS Divider End Register 1 (CGM_PCS_DIVE1)	Section 28.3.1.3
0x010C	PCS Divider Start Register 1 (CGM_PCS_DIVS1)	Section 28.3.1.4
0x0110	PCS Divider Change Register 2 (CGM_PCS_DIVC2)	Section 28.3.1.5
0x0114	PCS Divider End Register 2 (CGM_PCS_DIVE2)	Section 28.3.1.6
0x0118	PCS Divider Start Register 2 (CGM_PCS_DIVS2)	Section 28.3.1.7
0x011C–0x0127	Reserved	
0x0128	PCS Divider Change Register 4 (CGM_PCS_DIVC4)	Section 28.3.1.8
0x012C	PCS Divider End Register 4 (CGM_PCS_DIVE4)	Section 28.3.1.9
0x0130	PCS Divider Start Register 4 (CGM_PCS_DIVS4)	Section 28.3.1.10
0x0134–0x01CF	Reserved	
0x01D0	System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)	Section 28.3.1.11

Table 332. MC_CGM memory map (continued)

Address offset	Register	Location
0x01D4	Divider Update Type Register (CGM_DIV_UPD_TYPE)	Section 28.3.1.12
0x01D8	Divider Update Trigger Register (CGM_DIV_UPD_TRIG)	Section 28.3.1.13
0x01DC	Divider Update Status Register (CGM_DIV_UPD_STAT)	Section 28.3.1.14
0x01E4	System Clock Select Status Register (CGM_SC_SS)	Section 28.3.1.15
0x01E8	System Clock Divider 0 Configuration Register (CGM_SC_DC0)	Section 28.3.1.16
0x01EC	System Clock Divider 1 Configuration Register (CGM_SC_DC1)	Section 28.3.1.17
0x01F0	System Clock Divider 2 Configuration Register (CGM_SC_DC2)	Section 28.3.1.18
0x01F4	System Clock Divider 3 Configuration Register (CGM_SC_DC3)	Section 28.3.1.19
0x01F8	System Clock Divider 4 Configuration Register (CGM_SC_DC4)	Section 28.3.1.20
0x01FC–0x01FF	Reserved	
0x0200	Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)	Section 28.3.1.21
0x0204	Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)	Section 28.3.1.22
0x0208	Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)	Section 28.3.1.23
0x020C	Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)	Section 28.3.1.24
0x0210–0x021F	Reserved	
0x0220	Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)	Section 28.3.1.25
0x0224	Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)	Section 28.3.1.26
0x0228	Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)	Section 28.3.1.27
0x022C–0x025F	Reserved	
0x0260	Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)	Section 28.3.1.28
0x0264	Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)	Section 28.3.1.29
0x0268–0x027F	Reserved	
0x0280	Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)	Section 28.3.1.30
0x0284	Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)	Section 28.3.1.31
0x028C–0x02BF	Reserved	
0x02C0	Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)	Section 28.3.1.32
0x02C4	Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)	Section 28.3.1.33
0x02C8	Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)	Section 28.3.1.34
0x02CC–0x02FF	Reserved	
0x0300	Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)	Section 28.3.1.35
0x0304	Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)	Section 28.3.1.36
0x0308	Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)	Section 28.3.1.37
0x030C–0x031F	Reserved	
0x0320	Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)	Section 28.3.1.38

Table 332. MC_CGM memory map (continued)

Address offset	Register	Location
0x0324	Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)	*Section 28.3.1.39*
0x0328	Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)	*Section 28.3.1.40*
0x032C–0x035F	Reserved	
0x0360	Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)	*Section 28.3.1.41*
0x0364	Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)	*Section 28.3.1.42*
0x0368	Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)	*Section 28.3.1.43*
0x036C–0x037F	Reserved	
0x0380	Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)	*Section 28.3.1.44*
0x0384	Auxiliary Clock 12 Select Status Register (CGM_AC12_SS)	*Section 28.3.1.45*
0x0388	Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)	*Section 28.3.1.46*
0x038C	Auxiliary Clock 12 Divider 1 Configuration Register (CGM_AC12_DC1)	*Section 28.3.1.47*
0x0390	Auxiliary Clock 12 Divider 2 Configuration Register (CGM_AC12_DC2)	*Section 28.3.1.48*
0x0394	Auxiliary Clock 12 Divider 3 Configuration Register (CGM_AC12_DC3)	*Section 28.3.1.49*
0x0398–0x039B	Reserved	

Note: Any access to unused registers as well as write accesses to read-only registers will:

- *not change register content*
- *cause a transfer error*

28.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0104, as a half-word at address offset 0x0106, or as a byte at address offset 0x0107.

28.3.1.1 PCS Switch Duration Register (CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See *Section 28.4.1.2: Progressive system clock switching*, for details on how to set this value. This register is reset only on a power-on reset.

Offset: 0x0100 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R/W	SDUR							
POR	0	0	0	0	0	0	0	0

Figure 246. PCS Switch Duration Register (CGM_PCS_SDUR)

Table 333. CGM_PCS_SDUR field descriptions

Field	Description
SDUR	Switch Duration This value defines the duration of one PCS clock switch step in terms of 16 MHz int. RC osc. cycles.

28.3.1.2 PCS Divider Change Register 1 (CGM_PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the external crystal osc. on ramp-up and ramp-down, respectively. See *Section 28.4.1.2: Progressive system clock switching*, for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset : 0x0104 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	INIT															
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 247. PCS Divider Change Register 1 (CGM_PCS_DIVC1)

Table 334. CGM_PCS_DIVC1 field descriptions

Field	Description
INIT	**Divider Change Initial Value** This is initial change value of the clock divider for the clock ramp-up phase when switching to the external crystal osc..
RATE	**Divider Change Rate** This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the external crystal osc..

28.3.1.3 PCS Divider End Register 1 (CGM_PCS_DIVE1)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the external crystal osc. on ramp-down. See *Section 28.4.1.2: Progressive system clock switching*, for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0108 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 248. PCS Divider End Register 1 (CGM_PCS_DIVE1)

Table 335. CGM_PCS_DIVE1 field descriptions

Field	Description
DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the external crystal osc..

28.3.1.4 PCS Divider Start Register 1 (CGM_PCS_DIVS1)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the external crystal osc. on ramp-up. Register *n* corresponds to system clock source *n*. See *Section 28.4.1.2: Progressive system clock switching*, for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x010C Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	DIVS															
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 249. PCS Divider Start Register 1 (CGM_PCS_DIVS1)

Table 336. CGM_PCS_DIVS1 field descriptions

Field	Description
DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the external crystal osc..

28.3.1.5 PCS Divider Change Register 2 (CGM_PCS_DIVC2)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL0 PHI on ramp-up and ramp-down, respectively. See *Section 28.4.1.2: Progressive system clock switching*, for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0110 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	INIT															
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 250. PCS Divider Change Register 2 (CGM_PCS_DIVC2)

Table 337. CGM_PCS_DIVC2 field descriptions

Field	Description
INIT	**Divider Change Initial Value** This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.
RATE	**Divider Change Rate** This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL0 PHI.

28.3.1.6 PCS Divider End Register 2 (CGM_PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL0 PHI on ramp-down. See *Section 28.4.1.2: Progressive system clock switching* for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0114 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 251. PCS Divider End Register 2 (CGM_PCS_DIVE2)

Table 338. CGM_PCS_DIVE2 field descriptions

Field	Description
DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the PLL0 PHI.

28.3.1.7 PCS Divider Start Register 2 (CGM_PCS_DIVS2)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL0 PHI on ramp-up. See *Section 28.4.1.2: Progressive system clock switching*, for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0118 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 252. PCS Divider Start Register 2 (CGM_PCS_DIVS2)

Table 339. CGM_PCS_DIVS2 field descriptions

Field	Description
DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.

28.3.1.8 PCS Divider Change Register 4 (CGM_PCS_DIVC4)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL1 PHI on ramp-up and ramp-down, respectively. See *Section 28.4.1.2: Progressive system clock switching*, for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0128 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INIT															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 253. PCS Divider Change Register 4 (CGM_PCS_DIVC4)

Table 340. CGM_PCS_DIVC4 field descriptions

Field	Description
INIT	**Divider Change Initial Value** This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.
RATE	**Divider Change Rate** This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL1 PHI.

28.3.1.9 PCS Divider End Register 4 (CGM_PCS_DIVE4)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL1 PHI on ramp-down. See *Section 28.4.1.2: Progressive system clock switching*, for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x012C Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 254. PCS Divider End Register 4 (CGM_PCS_DIVE4)

Table 341. CGM_PCS_DIVE4 field descriptions

Field	Description
DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the PLL1 PHI.

28.3.1.10 PCS Divider Start Register 4 (CGM_PCS_DIVS4)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL1 PHI on ramp-up. See *Section 28.4.1.2: Progressive system clock switching* for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0130 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	DIVS															
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 255. PCS Divider Start Register 4 (CGM_PCS_DIVS4)

Table 342. CGM_PCS_DIVS4 field descriptions

Field	Description
DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.

28.3.1.11 System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)

This register selects whether the system clock divider ratios will change from the current configuration to the configuration in the next update. If the value of SYS_DIV_RATIO_CHNG is ‘1’ at the time of a write to the CGM_DIV_UPD_TRIG register and a pre-loaded system clock divider update is pending, the crossbar is halted while the system clock divider update takes place. If the value of SYS_DIV_RATIO_CHNG is ‘0’, the crossbar switch is not halted. This register has no effect if the system clock divider update is configured in the CGM_DIV_UPD_TYPE register to be immediate.

An example of a divider ratio not changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 4. The ratio is 1:2 in both configurations. An example of a divider ratio changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 6. The ratio changes from 1:2 to 1:3. The value of SYS_DIV_RATIO_CHNG is returned to ‘1’ automatically on each write to the CGM_DIV_UPD_TRIG register.

Caution: Software must never write a ‘0’ to the SYS_DIV_RATIO_CHNG bit if the system clock divider ratios change from the current configuration to the next configuration. This can potentially cause errors in the crossbar which lead to lost data.

Offset: 0x01D0 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SYS_DIV_RATIO_CHNG
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 256. System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)

Table 343. CGM_SC_DIV_RC field descriptions

Field	Description
SYS_DIV_ RATIO_CHNG	System Clock Divider Ratio Change 0 The system clock divider ratios does not change with the next system clock divider configuration update 1 The system clock divider ratios changes with the next system clock divider configuration update

28.3.1.12 Divider Update Type Register (CGM_DIV_UPD_TYPE)

This register selects whether the dividers associated with a clock are updated immediately on writing to the corresponding divider configuration register or only on writing to the divider update trigger register CGM_DIV_UPD_TRIG, after which the pre-loaded configuration in the divider configuration registers will take effect.

Offset : 0x01D4 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_UPD_TYPE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 257. Divider Update Type Register (CGM_DIV_UPD_TYPE)

Table 344. CGM_DIV_UPD_TYPE field descriptions

Field	Description
SYS_UPD_TYPE	System Clock Divider Update Type 0 The configuration for each system clock divider is updated immediately on writing to the corresponding CGM_SC_DCi register 1 The configuration for each system clock divider is pre-loaded on writing to the corresponding CGM_SC_DCi register, and the pre-loaded configurations of all the system clock dividers are updated on writing to the CGM_DIV_UPD_TRIG register

28.3.1.13 Divider Update Trigger Register (CGM_DIV_UPD_TRIG)

Writing any value to this register will cause all dividers that have corresponding bits set to '1' in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

Offset: 0x01D8 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	DIV_UPD_TRIGGER															
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	DIV_UPD_TRIGGER															
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 258. Divider Update Trigger Register (CGM_DIV_UPD_TRIG)

Table 345. CGM_DIV_UPD_TRIG field descriptions

Field	Description
DIV_UPD_TRIGGER	Writing any value to this register causes all dividers that have corresponding bits set to '1' in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register always returns all zeroes.

28.3.1.14 Divider Update Status Register (CGM_DIV_UPD_STAT)

This register indicates whether a divider for each clock is in the process of being updated. If the CGM_DIV_UPD_TYPE[i] is '0', CGM_DIV_UPD_STAT[i] is set to '1' immediately on writing to one of the corresponding divider configuration registers. Otherwise, CGM_DIV_UPD_STAT[i] is set to '1' immediately on writing to the CGM_DIV_UPD_TRIG register if one of the corresponding divider configurations has been pre-loaded. CGM_DIV_UPD_STAT[i] is cleared to '0' as soon as all the corresponding dividers have been updated and the dividers generate the desired frequency.

Offset: 0x01DC Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_UPD_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 259. Divider Update Status Register (CGM_DIV_UPD_STAT)

Table 346. CGM_DIV_UPD_STAT field descriptions

Field	Description
SYS_UPD_STAT	System Clock Divider Update Status 0 The configuration for at least one system clock divider is not being updated 1 The configuration for at least one system clock divider is being updated

28.3.1.15 System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Offset: 0x01E4 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	SWTRG			SWIP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 260. System Clock Select Status Register (CGM_SC_SS)

Table 347. CGM_SC_SS field descriptions

Field	Description
SELSTAT	**System Clock Source Selection Status** This value indicates the current source for the system clock. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 PLL1 PHI 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 system clock is disabled
SWTRG	Switch Trigger cause This value indicates the cause for the latest clock source switch. 000 Reserved 001 switch after request from MC_ME succeeded 010 switch after request from MC_ME failed due inactive target clock 011 switch after request from MC_ME failed due inactive current clock 100 switch to 16 MHz int. RC osc. due to SAFE mode request or reset succeeded 101 switch to 16 MHz int. RC osc. due to SAFE mode request or reset, but current clock source was inactive 110 Reserved 111 Reserved
SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress

28.3.1.16 System Clock Divider 0 Configuration Register (CGM_SC_DC0)

This register controls system clock divider 0. See *Section 28.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x01E8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 261. System Clock Divider 0 Configuration Register (CGM_SC_DC0)

Table 348. CGM_SC_DC0 field descriptions

Field	Description
DE	Divider 0 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
DIV	Divider 0 Division Value The resultant divider clock will have a period ‘DIV + 1’ times that of the system clock.

28.3.1.17 System Clock Divider 1 Configuration Register (CGM_SC_DC1)

This register controls system clock divider 1. See *Section 28.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x01EC Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 262. System Clock Divider 1 Configuration Register (CGM_SC_DC1)

Table 349. CGM_SC_DC1 field descriptions

Field	Description
DE	Divider 1 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
DIV	Divider 1 Division Value The resultant HPBM_CLK will have a period 'DIV + 1' times that of the system clock.

28.3.1.18 System Clock Divider 2 Configuration Register (CGM_SC_DC2)

This register controls system clock divider 2. See *Section 28.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x01F0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 263. System Clock Divider 2 Configuration Register (CGM_SC_DC2)

Table 350. CGM_SC_DC2 field descriptions

Field	Description
DE	Divider 2 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
DIV	Divider 2 Division Value The resultant PBRIDGE_CLK will have a period 'DIV + 1' times that of the system clock.

28.3.1.19 System Clock Divider 3 Configuration Register (CGM_SC_DC3)

This register controls system clock divider 3. See *Section 28.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x01F4 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 264. System Clock Divider 3 Configuration Register (CGM_SC_DC3)

Table 351. CGM_SC_DC3 field descriptions

Field	Description
DE	Divider 3 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
DIV	Divider 3 Division Value The resultant FBRIDGE_CLK will have a period ‘DIV + 1’ times that of the system clock.

28.3.1.20 System Clock Divider 4 Configuration Register (CGM_SC_DC4)

This register controls system clock divider 4. See *Section 28.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x01F8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 265. System Clock Divider 4 Configuration Register (CGM_SC_DC4)

Table 352. CGM_SC_DC4 field descriptions

Field	Description
DE	Divider 4 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
DIV	Divider 4 Division Value The resultant PFBRIDGE_CLK will have a period ‘DIV + 1’ times that of the system clock.

28.3.1.21 Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 0 divider 0: SARADC_CLK
- divided by auxiliary clock 0 divider 1: EMIOS_CLK

Offset: 0x0200 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 266. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

Table 353. CGM_AC0_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 0 Source Selection Control Selects the source for auxiliary clock 0. **Note:** Status of respective clock source must be ensured before selection. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.22 Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.

Offset: 0x0204 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 267. Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

Table 354. CGM_AC0_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 0 Source Selection Status** This value indicates the current source for auxiliary clock 0. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.23 Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

This register controls auxiliary clock 0 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset : 0x0208 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 268. Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

Table 355. CGM_AC0_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 0 divider 0 1 Enable auxiliary clock 0 divider 0
DIV	Divider Division Value The resultant SARADC_CLK will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SARADC_CLK remains disabled.

28.3.1.24 Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

This register controls auxiliary clock 0 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x020C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 269. Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

Table 356. CGM_AC0_DC1 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 0 divider 1 1 Enable auxiliary clock 0 divider 1
DIV	Divider Division Value The resultant EMIOS_CLK will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the EMIOS_CLK remains disabled.

28.3.1.25 Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 1 divider 0: LFAST_PLL

Offset: 0x0220 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 270. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

Table 357. CGM_AC1_SC field descriptions

Field	Description
SELCTL	**Auxiliary Clock 1 Source Selection Control** This value selects the current source for auxiliary clock 1. **Note:** Status of respective clock source must be ensured before selection. 0000 Reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 LFAST-SysClk pin 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.26 Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

This register provides the current auxiliary clock 1 source selection.

Offset : 0x0224 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 271. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

Table 358. CGM_AC1_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 1 Source Selection Status** This value indicates the current source for auxiliary clock 1. 0000 Reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 LFAST-SysClk pin 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.27 Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

This register controls auxiliary clock 1 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0228 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 272. Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

Table 359. CGM_AC1_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 1 divider 0 1 Enable auxiliary clock 1 divider 0
DIV	Divider Division Value The resultant LFAST_PLL will have a period 'DIV + 1' times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the LFAST_PLL remains disabled.

28.3.1.28 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

This register is used to select the current clock source for the PLL0 reference clock.

Offset: 0x0260 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 273. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

Table 360. CGM_AC3_SC field descriptions

Field	Description
SELCTL	**Auxiliary Clock 3 Source Selection Control** This value selects the current source for auxiliary clock 3. **Note:** Status of respective clock source must be ensured before selection. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.29 Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

Offset: 0x0264 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 274. Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

Table 361. CGM_AC3_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 3 Source Selection Status** This value indicates the current source for auxiliary clock 3. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.30 Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

This register is used to select the current clock source for the PLL1 reference clock.

Offset: 0x0280 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 275. Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

Table 362. CGM_AC4_SC field descriptions

Field	Description
SELCTL	**Auxiliary Clock 4 Source Selection Control** This value selects the current source for auxiliary clock 4. **Note:** Status of respective clock source must be ensured before selection. 0000 Reserved 0001 external crystal osc. 0010 Reserved 0011 PLL0 PHI1 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.31 Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

This register provides the current auxiliary clock 4 source selection.

Offset : 0x0284 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 276. Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

Table 363. CGM_AC4_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 4 Source Selection Status** This value indicates the current source for auxiliary clock 4. 0000 Reserved 0001 external crystal osc. 0010 Reserved 0011 PLL0 PHI1 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.32 Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 6 divider 0: SYSCLK0 pin clock

Offset: 0x02C0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 277. Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)

Table 364. CGM_AC6_SC field descriptions

Field	Description
SELCTL	**Auxiliary Clock 6 Source Selection Control** This value selects the current source for auxiliary clock 6. **Note:** Status of respective clock source must be ensured before selection. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 PLL1 PHI 0101 Reserved 0110 rtc_clk 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.33 Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)

This register provides the current auxiliary clock 6 source selection.

Offset: 0x02C4 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 278. Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)

Table 365. CGM_AC6_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 6 Source Selection Status** This value indicates the current source for auxiliary clock 6. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 PLL1 PHI 0101 Reserved 0110 rtc_clk 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.34 Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)

This register controls auxiliary clock 6 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x02C8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 279. Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)

Table 366. CGM_AC6_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 6 divider 0 1 Enable auxiliary clock 6 divider 0
DIV	Divider Division Value The resultant SYSCLK0 pin clock will have a period 'DIV + 1' times that of auxiliary clock 6. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SYSCLK0 pin clock remains disabled.

28.3.1.35 Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 8 divider 0: CAN0 clock

Offset: 0x0300 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 280. Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

Table 367. CGM_AC8_SC field descriptions

Field	Description
SELCTL	**Auxiliary Clock 8 Source Selection Control** This value selects the current source for auxiliary clock 8. **Note:** Status of respective clock source must be ensured before selection. 0000 Reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.36 Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

This register provides the current auxiliary clock 8 source selection.

Offset: 0x0304 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 281. Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

Table 368. CGM_AC8_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 8 Source Selection Status** This value indicates the current source for auxiliary clock 8. 0000 Reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.37 Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

This register controls auxiliary clock 8 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0308 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 282. Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

Table 369. CGM_AC8_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 8 divider 0 1 Enable auxiliary clock 8 divider 0
DIV	Divider Division Value The resultant CAN0 clock will have a period 'DIV + 1' times that of auxiliary clock 8. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CAN0 clock remains disabled.

28.3.1.38 Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 9 divider 0: RTI_CLK

Offset: 0x0320 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 283. Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)

Table 370. CGM_AC9_SC field descriptions

Field	Description
SELCTL	**Auxiliary Clock 9 Source Selection Control** This value selects the current source for auxiliary clock 9. **Note:** Status of respective clock source must be ensured before selection. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.39 Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)

This register provides the current auxiliary clock 9 source selection.

Offset: 0x0324 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 284. Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)

Table 371. CGM_AC9_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 9 Source Selection Status** This value indicates the current source for auxiliary clock 9. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.40 Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)

This register controls auxiliary clock 9 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0328 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 285. Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)

Table 372. CGM_AC9_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 9 divider 0 1 Enable auxiliary clock 9 divider 0
DIV	Divider Division Value The resultant RTI_CLK will have a period 'DIV + 1' times that of auxiliary clock 9. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the RTI_CLK remains disabled.

28.3.1.41 Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 11 divider 0: CAN1 clock

Offset: 0x0360 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 286. Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)

Table 373. CGM_AC11_SC field descriptions

Field	Description
SELCTL	**Auxiliary Clock 1 Source Selection Control** This value selects the current source for auxiliary clock 1. **Note:** Status of respective clock source must be ensured before selection. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 Reserved 0110 reserved 0111 reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.42 Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)

This register provides the current auxiliary clock 11 source selection.

Offset: 0x0364 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 287. Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)

Table 374. CGM_AC11_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 11 Source Selection Status** This value indicates the current source for auxiliary clock 11. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 Reserved 0110 reserved 0111 reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.43 Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)

This register controls auxiliary clock 11 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0368 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 288. Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)

Table 375. CGM_AC11_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 11 divider 0 1 Enable auxiliary clock 11 divider 0
DIV	Divider Division Value The resultant CAN1 clock will have a period 'DIV + 1' times that of auxiliary clock 11. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CAN1 clock remains disabled.

28.3.1.44 Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 12 divider 0: DSPI_CLK0
- divided by auxiliary clock 12 divider 1: DSPI_CLK1
- divided by auxiliary clock 12 divider 2: LIN_CLK
- divided by auxiliary clock 12 divider 3: PER_CLK0

Offset: 0x0380 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 289. Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)

Table 376. CGM_AC12_SC field descriptions

Field	Description
SELCTL	Auxiliary Clock 12 Source Selection Control Selects the source for auxiliary clock 12. **Note:** Status of respective clock source must be ensured before selection. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.45 Auxiliary Clock 12 Select Status Register (CGM_AC12_SS)

This register provides the current auxiliary clock 12 source selection.

Offset: 0x0384 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 290. Auxiliary Clock 12 Select Status Register (CGM_AC12_SS)

Table 377. CGM_AC12_SS field descriptions

Field	Description
SELSTAT	**Auxiliary Clock 12 Source Selection Status** This value indicates the current source for auxiliary clock 12. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

28.3.1.46 Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)

This register controls auxiliary clock 12 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Offset: 0x0388 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV_FMT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 291. Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)

Table 378. CGM_AC12_DC0 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 12 divider 0 1 Enable auxiliary clock 12 divider 0
DIV	Divider Division Value The resultant DSPI_CLK0 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the DSPI_CLK0 remains disabled.
DIV_FMT	Divider Value Format - This value indicates the format of the division value for calculating the division factor 00 Division factor = (DIV + 1) / 1 01 Division factor = (DIV + 1) / 10 10 Division factor = (DIV + 1) / 100 11 Division factor = (DIV + 1) / 1,000

28.3.1.47 Auxiliary Clock 12 Divider 1 Configuration Register (CGM_AC12_DC1)

This register controls auxiliary clock 12 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x038C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 292. Auxiliary Clock 12 Divider 1 Configuration Register (CGM_AC12_DC1)

Table 379. CGM_AC12_DC1 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 12 divider 1 1 Enable auxiliary clock 12 divider 1
DIV	Divider Division Value The resultant DSPI_CLK1 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the DSPI_CLK1 remains disabled.

28.3.1.48 Auxiliary Clock 12 Divider 2 Configuration Register (CGM_AC12_DC2)

This register controls auxiliary clock 12 divider 2.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0390 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 293. Auxiliary Clock 12 Divider 2 Configuration Register (CGM_AC12_DC2)

Table 380. CGM_AC12_DC2 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 12 divider 2 1 Enable auxiliary clock 12 divider 2
DIV	Divider Division Value The resultant LIN_CLK will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the LIN_CLK remains disabled.

28.3.1.49 Auxiliary Clock 12 Divider 3 Configuration Register (CGM_AC12_DC3)

This register controls auxiliary clock 12 divider 3.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Offset: 0x0394 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 294. Auxiliary Clock 12 Divider 3 Configuration Register (CGM_AC12_DC3)

Table 381. CGM_AC12_DC3 field descriptions

Field	Description
DE	Divider Enable 0 Disable auxiliary clock 12 divider 3 1 Enable auxiliary clock 12 divider 3
DIV	Divider Division Value The resultant PER_CLK0 will have a period ‘DIV + 1’ times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the PER_CLK0 remains disabled.

28.4 Functional description

28.4.1 System clock generation

Figure 220: Clock generation shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the 16 MHz int. RC osc. as the system clock and to ignore the system clock select.

28.4.1.1 System clock source selection

During normal operation, the system clock selection is controlled

- on a SAFE mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

28.4.1.2 Progressive system clock switching

In order to prevent sudden voltage drops and overshoots due to frequency and load changes, the MC_ME requests the MC_CGM to ramp the system clock frequency down and/or up based on the power level values of the current and target modes.

During ramp-down, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVC*n*, and CGM_PCS_DIVE*n* registers, where *n* corresponds to the current system clock source selection.

During ramp-up, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVC*n*, and CGM_PCS_DIVS*n* registers, where n corresponds to the target system clock source selection.

28.4.1.2.1 Generic clock change requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in *Equation 14:* .

Equation 14

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

28.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field CGM_PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of 16 MHz int. RC osc.. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

28.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the PCS_DIVC*n*[RATE] register is given in *Table 382*.

Table 382. MC_CGM CGM_PCS_DIVC*n*[RATE] values

a_{max}	d	CGM_PCS_DIVCn[RATE]
0.05	0.012	12
0.10	0.048	48
0.15	0.112	112
0.20	0.184	184

28.4.1.2.4 Generic clock change properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in *Equation 15:* .

Equation 15

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d}} \right\rceil$$

28.4.1.2.5 Clock ramp-down

The clock ramp-down starts with the divider value 1 and with the given divider increment value PCD_DIVCn[RATE] and ends with the given divider value PCS_DIVEn[DIVE].

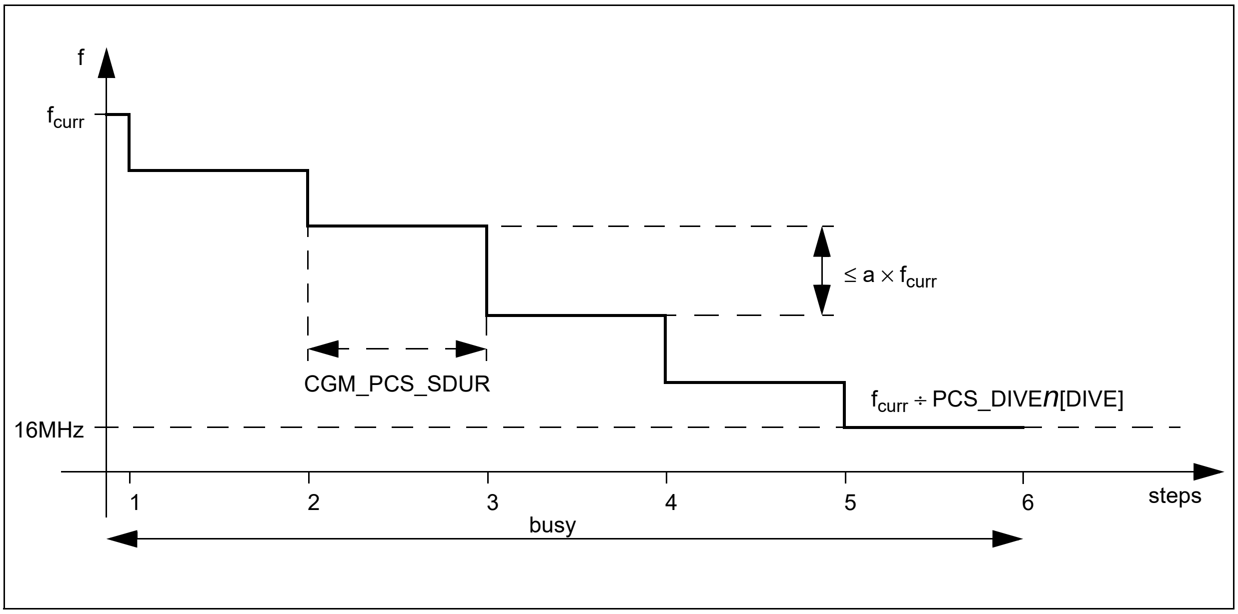
The divider end value for clock ramp-down is given in *Equation 16:* .

Equation 16

$$\mathbf{PCS\_DIVEn[DIVE]} = \frac{f_{src}}{16MHz} \times 1000 - 1$$

Where f_{curr} is the frequency of the currently selected system clock source.

Figure 295. MC_CGM System Clock Ramp-Down Timing (k = 6 example)



28.4.1.2.6 Clock ramp-up

The clock ramp-up starts with the given divider value PCS_DIVSn[DIVS] and with the given divider decrement value PCD_DIVCn[INIT] and ends with the divider value 1.

The initial divider change start value PCS_DIVCn[INIT] for system clock ramp-up is given in *Equation 17:* .

Equation 17

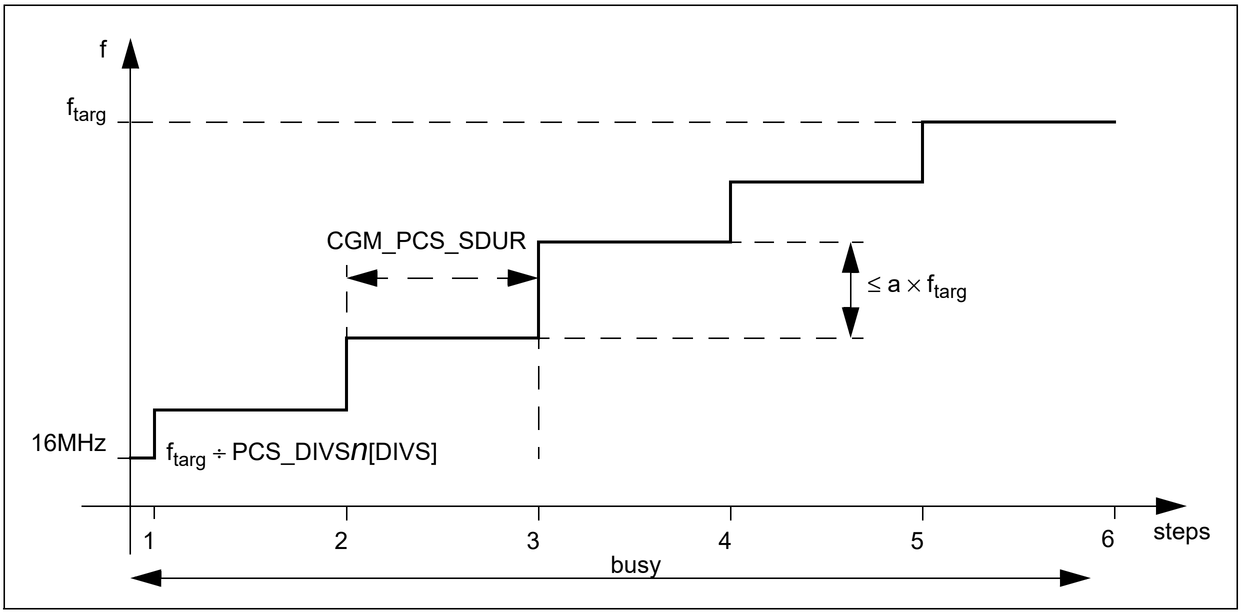
$$\mathbf{PCS\_DIVCn[INIT]} = d \times k \times 1000 - 1$$

Where k is calculated from *Equation 15:* using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in *Equation 18:* .

Equation 18

$$\textbf{PCS\_DIVSn[DIVS]} = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

Figure 296. MC_CGM System Clock Ramp-Up Timing (k = 6 example)



28.4.1.3 System clock disable

During the TEST mode, the system clock can be disabled by the MC_ME.

28.4.1.4 System clock dividers

The MC_CGM generates the following derived clocks from the system clock:

- CORE_CLK (Core_2), HSM_CLK, XBAR_CLK - controlled by the CGM_SC_DC0 register
- HPBM_CLK - controlled by the CGM_SC_DC1 register
- PBRIDGE_CLK - controlled by the CGM_SC_DC2 register
- FBRIDGE_CLK - controlled by the CGM_SC_DC3 register
- PFBRIDGE_CLK - controlled by the CGM_SC_DC4 register

28.4.1.4.1 System clock divider synchronization

The system clock dividers are synchronized to each other such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This, however, imposes limitations on the division factors which can be used. In order for the synchronization to work properly, each division factor must be selected such that its value is an integer multiple of each division factor that has a lower value.

The allowed combinations of division factors is given in *Table 306: SYS_CLK divisor clock ratios at power up*.

28.4.1.4.2 System clock divider update triggering

As opposed to the auxiliary clock dividers, which update immediately on a configuration change, the system clock dividers are updated only with the next software triggered mode change. This is to ensure that the system clock dividers are updated simultaneously and that no intermediate inconsistent configurations can occur. It also ensures that the divider configurations are aligned with the target mode’s system clock selection.

The system clock divider configuration update sequence is as follows:

1. Configure the target mode in the MC_ME to select the desired system clock source.
2. Configure the CGM_SC_DC0-4 registers as desired and aligned with the target system clock source frequency making sure that the dividers are configured consistently as described in *Section 28.4.1.4.1: System clock divider synchronization*.
3. Request a mode change in the MC_ME to the desired target mode.
4. When mode change has completed and the system clock switching has been successful, the new system clock divider configuration will be in effect.

See *Chapter 56: Mode Entry Module (MC_ME)*, for details on how to configure modes and make mode change requests.

28.4.2 Auxiliary clock generation

Figure 220: Clock generation shows the block diagrams of the generation logic for the various auxiliary clocks.

See the following sections for auxiliary clock selection control:

- *Section 28.3.1.21: Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)*
- *Section 28.3.1.25: Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)*
- *Section 28.3.1.28: Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)*
- *Section 28.3.1.30: Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)*
- *Section 28.3.1.32: Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)*
- *Section 28.3.1.35: Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)*
- *Section 28.3.1.38: Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)*
- *Section 28.3.1.41: Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)*
- *Section 28.3.1.44: Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)*

28.4.2.1 Auxiliary clock dividers

The MC_CGM generates the following derived clocks:

- SARADC_CLK - controlled by the CGM_AC0_DC0 register
- EMIOS_CLK - controlled by the CGM_AC0_DC1 register
- LFAST_PLL - controlled by the CGM_AC1_DC0 register
- SYSCLK0 pin clock - controlled by the CGM_AC6_DC0 register
- CAN0 clock - controlled by the CGM_AC8_DC0 register
- RTI_CLK - controlled by the CGM_AC9_DC0 register
- CAN1 clock - controlled by the CGM_AC11_DC0 register
- DSPI_CLK0 - controlled by the CGM_AC12_DC0 register
- DSPI_CLK1 - controlled by the CGM_AC12_DC1 register
- LIN_CLK - controlled by the CGM_AC12_DC2 register
- PER_CLK0 - controlled by the CGM_AC12_DC3 register

28.4.3 Dividers functional description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- System clock divider configuration registers
  - *Section 28.3.1.16: System Clock Divider 0 Configuration Register (CGM_SC_DC0)*
  - *Section 28.3.1.17: System Clock Divider 1 Configuration Register (CGM_SC_DC1)*
  - *Section 28.3.1.18: System Clock Divider 2 Configuration Register (CGM_SC_DC2)*
  - *Section 28.3.1.19: System Clock Divider 3 Configuration Register (CGM_SC_DC3)*
  - *Section 28.3.1.20: System Clock Divider 4 Configuration Register (CGM_SC_DC4)*
- Auxiliary clock divider configuration registers
  - *Section 28.3.1.23: Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)*
  - *Section 28.3.1.24: Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)*
  - *Section 28.3.1.27: Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)*
  - *Section 28.3.1.34: Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)*
  - *Section 28.3.1.37: Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)*
  - *Section 28.3.1.40: Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)*
  - *Section 28.3.1.43: Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)*
  - *Section 28.3.1.46: Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)*
  - *Section 28.3.1.47: Auxiliary Clock 12 Divider 1 Configuration Register (CGM_AC12_DC1)*
  - *Section 28.3.1.48: Auxiliary Clock 12 Divider 2 Configuration Register (CGM_AC12_DC2)*
  - *Section 28.3.1.49: Auxiliary Clock 12 Divider 3 Configuration Register (CGM_AC12_DC3)*

The reset value of all counters is ‘1’. If a divider has its DE bit in the respective configuration register set to ‘0’ (the divider is disabled), any value in its DIV field is ignored.

28.4.3.1 Fractional divider details

The clock divider divides the selected clock source frequency by a fractional value given by the DIV and DIV_FMT fields of the CGM_AC register.

Equation 19

$$F_{\text{divided clock}} = F_{\text{source clock}} \div ((\mathbf{DIV} + 1) \div 10^{\mathbf{DIV\_FMT}})$$

The divided clock frequency is actually an average frequency, since the divider uses only the available rising and falling edges of the source clock. Therefore, the actual frequency switches between two frequencies which are the two frequencies closest to the desired frequency (above and below) which can be extracted by a division factor *n*/2, where *n* is an integer greater than 1. The switching between the two frequencies is such that the resultant clock's rising edge never deviates from the ideal clock's rising edge by more than 25% of the source clock period and that the resultant average frequency over DIV + 1 source clock periods (= 10^{DIV_FMT} divided clock periods) is the desired frequency.

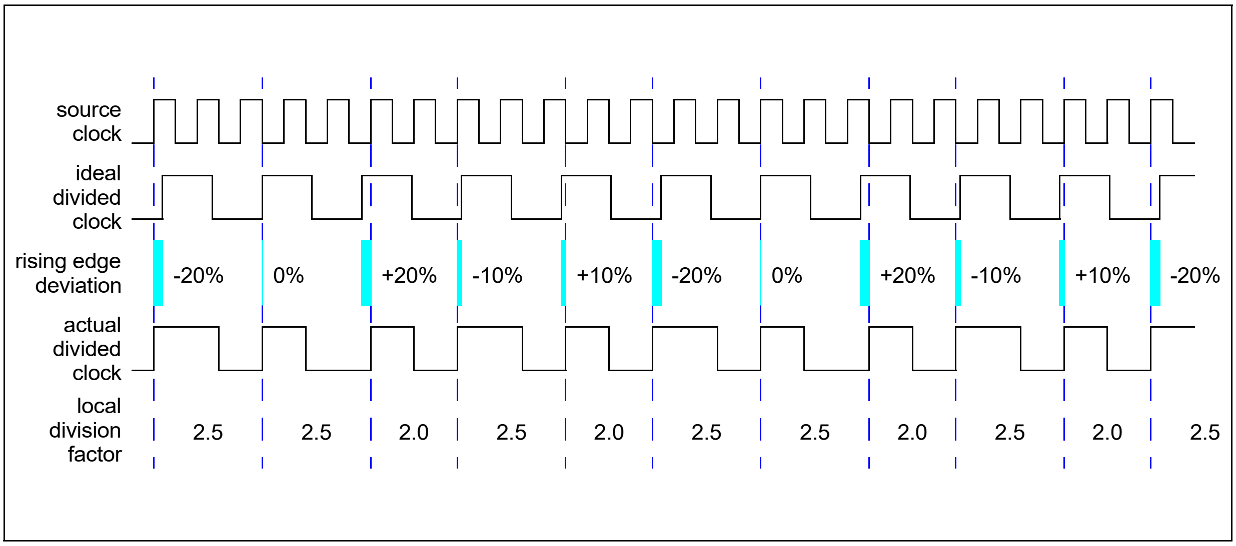
Example 1 Example: Fractional clock division

Desired division factor = 2.3

DIV = 22

DIV_FMT = 1

Figure 297. MC_CGM fractional division example waveform (divide by 2.3)



Equation 20

$$F_{\text{divided clock}}\big|_{\text{max}} = F_{\text{source clock}} \div 2.0$$

Equation 21

$$F_{\text{divided clock}}\big|_{min} = F_{\text{source clock}} \div 2.5$$

Equation 22

$$F_{\text{divided clock}}\big|_{ave} = F_{\text{source clock}} \div \frac{((2.5 \times 6) + (2.0 \times 4))}{10}$$

Therefore,

Equation 23

$$F_{\text{divided clock}}\big|_{ave} = F_{\text{source clock}} \div 2.3$$

28.4.3.2 Divider Update Triggering

There are two methods to update the dividers associated with the system clock and each auxiliary clock: immediate and register triggered. Which method is used for which set of dividers is controlled by the CGM_DIV_UPD_TYPE register.

Regardless of which method is selected for a given group of dividers, once an update of one of the dividers of that group has been triggered, the associated bit in the CGM_UPD_STAT register is set. This bit is then cleared when all the dividers of that group have completed the update process and are running with the current configuration.

28.4.3.2.1 Immediate Divider Update

If a divider group has been configured to be updated immediately (such as the corresponding CGM_DIV_UPD_TYPE register bit = ‘0’), the configuration of a divider in that group will be applied immediately on the write to that divider’s configuration register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

28.4.3.2.2 Software-triggered divider update

If a divider group has been configured to be updated via a software trigger (such as the corresponding CGM_DIV_UPD_TYPE register bit = ‘1’), the configuration of a divider in that group will not be applied immediately on the write to that divider’s configuration register. Instead, the configuration in the corresponding configuration register will be applied only when software writes any value to the CGM_DIV_UPD_TRIG register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

In the case of the system clock dividers, if the SYS_DIV_RATIO_CHNG bit in the CGM_SC_DIV_RC register = ‘1’, the crossbar switch is requested to halt its ongoing transactions, and only after the crossbar switch has acknowledged the request does the update of the system clock dividers take place. Once these updates have completed, the halt request to the crossbar is deasserted.

Note: If halting the crossbar can cause a bus master to starve and thus lead to lost data or other problems, it is recommended to vary the system clock divider configurations without changing the ratios and to set CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] to ‘0’.

In order to ensure a clean clock divider configuration update, software must check that all ongoing updates have already completed (such as all relevant bits in the CGM_DIV_UPD_STAT register are ‘0’) before triggering a new update via the CGM_DIV_UPD_TRIG register.

29 OSC digital interface (XOSC)

29.1 Introduction

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provides the register interface for the programmable features. Selection of the XOSC as a source clock is controlled by the Clock Generation module (MC_CGM). The XOSC is enabled and disabled by configuring the Mode Entry module (MC_ME).

The programmable features of the XOSC digital interface are as follows:

- Oscillator power-down control and status
- Oscillator startup delay
- Oscillator clock available interrupt
- Oscillator bypass mode control

29.2 Functional description

The XOSC digital interface provides control of the on-chip oscillator and the register interface for the mode and startup delay control. The oscillator provides an output clock that can used as an input to PLL(s) and can be selected as the reference clock for many of the peripherals on the device.

Table 383 shows the possible configurations of the XOSC.

Table 383. XOSC configurations

OSC EN	XOSC_CTL[OSCBYP]	XTAL	EXTAL	XOSC output	XOSC mode
0	0	No crystal, high-impedance	No crystal, high-impedance	0	Power down, IDDQ
0	1	x	external clock	Undefined	Reserved, OSC Disabled
1	1	x	external clock	XTALOUT	Bypass, OSC Enabled
1	0	crystal	crystal	XTALOUT	Normal, OSC Enabled

29.2.1 Oscillator power-down control and status

Enabling and disabling of the oscillator is done in the Mode Entry (MC_ME) module (see “Mode Entry Module (MC_ME)” chapter). The ME_<*mode*>_MC[XOSCON] controls the power-down of oscillator based on the current device mode (see “MC_ME Memory Map” table for available *modes*). The status of the XOSC is shown in the MC_ME_GS[S_XOSC].

29.2.2 Oscillator startup delay

A bit in the UTEST Flash memory determines whether the oscillator is enabled at power-up of the device (if the DCF client miscellaneous has been programmed, otherwise the valid configuration is the value present in *Chapter 9: Device Configuration Format (DCF) Records*, *Table 107: DCF client list*). If enabled, the oscillator is started after Phase3 of the device reset sequence. In order to allow the oscillator to reach full amplitude before use, the internal counter (OSCCNT) is started when the device exits reset. The counter is driven by the oscillator output clock. When the counter reaches XOSC_CTL[EOCV] × 512, a signal is sent to the MC_ME module, allowing a mode switch to the oscillator (when used as a system clock or an input to PLL(s)).

The default value for this field after reset depends on implementation and can be found in the XOSC section of the Device Configuration chapter. After reset, software can clear or change the EOCV field in order to shorten or lengthen the delay until the oscillator is ready.

Note: It is advised that if the EOCV value is modified it should be done as early as possible in user code. This will ensure that the new value will be used before OSCCNT reaches the count value.

29.2.3 Oscillator clock available interrupt

An interrupt can be generated when the count value is reached (OSCCNT = XOSC_CTL[EOCV] × 512). The XOSC_CTL[I_OSC] will be set and an interrupt will be generated if the interrupt mask is set (XOSC_CTL[M_OSC] = 1).

29.2.4 Oscillator bypass mode

The oscillator circuit can be bypassed by setting XOSC_CTL[OSCBYP]. This field can only be set by software, and is only cleared by a system reset. In bypass mode, the oscillator is disabled, and the input clock on the EXTAL pins is level-shifted and driven to the logic on the device. The bypass configuration is independent of the power-down mode of the oscillator.

Note: The external oscillator must be running before the XOSC is turned off. If XOSC is being turned on, the status will be updated only after the clock starts toggling.

29.2.4.1 Automatic Level Control (ALC)

The function of the ALC block is to provide high drive current during oscillator start up, and reduce current after oscillation in order to reduce power, distortion, RFI; and to avoid overdriving the crystal. ALC block can be controlled via DCF by configuring the UTEST miscellaneous bit 17 [XOSC_ALC_DIS]. Comparing the configurations with ALC disabled respect the case with ALC enabled, there are the following attributes:

- Higher oscillation amplitudes at crystal
- Distorted oscillation with several harmonics
- Increased EMI
- Higher crystal drive level
- Less susceptible to noise

The ALC enable configuration is suitable for most applications. The amplitude of oscillations is lower, hence less distortion in oscillations, less EMI, and also less crystal drive level. Feedback resistor is integrated thus, it does not require external resistors by offering a lower cost solution and simpler PCB design.

Note: It is recommended to make ALC feature enabled (that is, XOSC_ALC_DIS = 0) for low dissipation in crystal (crystal drive level).

Consult with crystal supplier for determining the need of extra resistance in XTAL to avoid crystal overdrive.

29.3 Memory map and register definition

Table 384 shows the memory map of the XOSC registers.

Table 384. XOSC memory map

Offset	Register	Location
0x00	XOSC Control Register (XOSC_CTL)	*Section 29.3.1.1*

29.3.1 Register descriptions

Note: Module generates transfer error (ips_xfr_err) only for 32 bytes address space (00 : 1F). If the address space allocated for this module exceeds 32 bytes, then transfer error has to be managed at SoC level for the remaining space.

29.3.1.1 XOSC Control register (XOSC_CTL)

Offset: 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OSCBYP(1)	0	0	0	0	0	0	0	EOCV(1)							
W																
Reset	–	0	0	0	0	0	0	0	–	–	–	–	–	–	–	–

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC(1)	0	0	0	0	0	0	0	I_OSC	0	0	0	0	0	0	0
W									w1c							
Reset	–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See the Clocking chapter for reset value.

Figure 298. XOSC Control register (XOSC_CTL)

Table 385. XOSC_CTL field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL is used as root clock.
8:15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV×512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.

Table 385. XOSC_CTL field descriptions (continued)

Field	Description
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
24 I_OSC	Crystal oscillator clock interrupt This bit can only be set by hardware when OSCCNT counter reaches the count value EOCV $\times$ 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.

30 IRCOSC digital interface

30.1 Introduction

The Internal RC Oscillator digital interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC)

It provides access to trimming information used for safety implementations.

30.2 Functional description

The IRCOSC provides a high frequency (f_{MRC}) clock with a nominal frequency of 16 MHz. After deassertion of reset, the output clock is available after a short time (see Data Sheet for timing information) which is required for stabilization. The IRCOSC status indication is found in the MC_ME_GS[S_IRC] field (see the “Mode Entry Module MC_ME” chapter).

During a power on reset sequence the following are initialized from TEST Flash memory:

- Native trimming bits
- Fine trimming bits

For safety reasons, these bits can be accessed through the IRCOSC Native Trimming register (NT). The main IRCOSC output frequency can be trimmed by configuring IRCOSC_CTL[USER_TRIM[5:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with δf_{var_SW}, as defined in the Data Sheet.

30.3 Memory map and register definition

Table 386 shows the memory map of the IRCOSC registers.

Table 386. IRCOSC memory map

Offset (hex)	Register	Width (in bits)	Access	Reset Value (hex)	Location
0x00	IRCOSC Control Register (IRCOSC_CTL)	32	R/W	0x0000_0000	Section 30.3.1.1
0x04	IRCOSC Native Trimming Register (IRCOSC_NT)	32	R	0xXX00_0XXX[(1)]	Section 30.3.1.2

1. See register section for specific reset values.

30.3.1 Register descriptions

30.3.1.1 IRCOSC control register (IRCOSC_CTL)

Offset: 0x0000 Access: Read/Write[1]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	USER_TRIM					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. IRCOSC_CTL is writable only in supervisor mode.

Figure 299. IRCOSC control register (IRCOSC_CTL)

Table 387. IRCOSC_CTL field descriptions

Field	Description
10:15 USER_TRIM	User trimming bits TRIM_BIT_USER[5:0] with respect to nominal factory frequency The MSB of the TRIM_BIT_USER bits is used to determine whether the frequency will be increased or decreased: – If MSB = 0 then a change in TRIM_BIT_USER[4:0] will decrease the frequency, and likewise, – if MSB = 1 then a change in TRIM_BIT_USER[4:0] will increase the frequency. TRIM_BIT_USER frequency trimming calculation: 110000 (-16) F1 + 5.16% 110001 (-15) F1 + 4.83% 110010 (-14) F1 + 4.50% 110011 (-13) F1 + 4.17% 110100 (-12) F1 + 3.84% 110101 (-11) F1 + 3.51% 110110 (-10) F1 + 3.19% 110111 (-9) F1 + 2.86% 111000 (-8) F1 + 2.54% 111001 (-7) F1 + 2.22% 111010 (-6) F1 + 1.89% 111011 (-5) F1 + 1.57% 111100 (-4) F1 + 1.25% 111101 (-3) F1 + 0.94% 111110 (-2) F1 + 0.62% 111111 (-1) F1 + 0.31% 000000 (0) F1 = 16 MHz 000001 (1) F1 - 0.29% 000010 (2) F1 - 0.58% 000011 (3) F1 - 0.87% 000100 (4) F1 - 1.17% 000101 (5) F1 - 1.45% 000110 (6) F1 - 1.74% 000111 (7) F1 - 2.02% 001000 (8) F1 - 2.30% 001001 (9) F1 - 2.59% 001010 (10) F1 - 2.87% 001011 (11) F1 - 3.15% 001100 (12) F1 - 3.43% 001101 (13) F1 - 3.71% 001110 (14) F1 - 3.98% 001111 (15) F1 - 4.26% 010000 (16) F1 - 4.53% All other values are reserved.

30.3.1.2 IRCOSC Native Trimming register (IRCOSC_NT)

The NT register contains trim values loaded from TEST Flash memory for the IRCOSC. Its status is enabled for voltage regulator and temperature sensor. This register is read only.

Offset: 0x0004 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FT_CODE						0	0	0	0	0	0	0	0
W																
Reset	0	0	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	REG_EN	0	0	0	RCTRIM							
W																
Reset	0	0	0	0	1	—(1)	0	0	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)

1. Reset value is coming from factory calibration (TEST Flash)

Figure 300. IRCOSC Native Trimming register (IRCOSC_NT)

Table 388. IRCOSC_NT field descriptions

Field	Description
2:7 FT_CODE	This bitfield indicates the trimming code applied to RCOSC analog macro based on temperature measured by vtsense
20 REG_EN	Enables voltage regulator: trimming value for safety check 0 Voltage regulator is not enabled 1 Voltage regulator is enabled
24:31 RCTRIM	IRCOSC trimming value: this bitfield reflects the coarse trim and fine trim applied to RC analog macro read from DCF clients.

31 LPRC digital interface

31.1 Introduction

The LPRC digital interface controls the internal Low Power RC oscillator (LPRC). It holds control and status registers accessible for application.

31.2 Low Power RC oscillator

The Low Power RC oscillator provides a low frequency clock of 1.024 MHz with a good stability with respect to supply and temperature variation.

Low Power RC oscillator is enabled during power on reset sequence and it is not expected that application switches off the LPRC oscillator (note that the software shall keep to 0 the MISC_CTRL_REG[LPRCREG_ENB] bit). In all modes the Low Power RC oscillator is always enabled. After switch on, the output clock is available only after 8.5 us which is required for stabilization. The clock source status is updated in the S_LPRC bit of LPRC_CTL register.

The LPRC clock can be further divided by a configurable division factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the LPRCDIV[4:0] bits of LPRC_CTL register.

31.3 Memory map and register definition

The LPRC registers are listed in *Table 389*.

Table 389. LPRC digital interface registers

Address offset (hex)	Register name	Location
0x0000	Low Power RC Control Register (CTL)	*Section 31.3.1.1*

31.3.1 Register description

31.3.1.1 Low Power RC Control register (CTL)

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	LPRCDIV					0	0	0	S_LPRC	0	0	0	0
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0

Figure 301. Low Power RC Control register (CTL)

Note: CTL register is writable only in supervisor mode.

Table 390. CTL register field descriptions

Field	Description
19:23 LPRCDIV	Low Power RC clock division factor These bits specify the Low Power RC oscillator output clock division factor. The output clock is divided by the factor LPRCDIV+1.
27 S_LPRC	Low Power RC clock status 0 LPRC is not providing a stable clock 1 LPRC is providing a stable clock

32 OSC32K digital interface (OSC32K)

32.1 Introduction

The OSC32K digital interface controls the operation of the 32 KHz slow external crystal oscillator. It holds control and status registers accessible for application.

The programmable features of the OSC32K digital interface are as follows:

- Oscillator power-down control and status
- Oscillator clock available interrupt
- Oscillator bypass mode control
- Output clock division factors ranging from 1 to 32

32.2 Memory map and register definition

Table 391 shows the memory map of the OSC32K registers.

Table 391. XOSC32K memory map

Offset	Register	Location
0x00	OSC32K Control Register (SXOSC_CTL)	*Section 32.2.1.1*

32.2.1 Register descriptions

32.2.1.1 OSC32K Control register (SXOSC_CTL)

Offset: 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OSCBYP[1]	0	0	0	0	0	0	0	EOCV							
W	OSCBYP[1]								EOCV							
Reset	–	0	0	0	0	0	0	0	–	–	–	–	–	–	–	–

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC	0	0	OSCDIV					I_OSC[2]	0	0	0	0	0	S_OSC	OSCON
W	M_OSC			OSCDIV					I_OSC[2]							OSCON
Reset	–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can read this field, and you can write a value of “1” to it. Writing a “0” has no effect. A reset will also clear this bit.
2. You can write a value of "0" or "1" to this field. However, writing a "1" will clear this field, and writing "0" will have no effect on the field value.

Figure 302. OSC32K Control register (SXOSC_CTL)

Table 392. SXOSC_CTL register field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock 1 EXTAL is used as root clock
8:15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV×512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked 1 Crystal oscillator clock interrupt is enabled
19:23 OSCDIV	Crystal oscillator clock division factor These bits specify the crystal oscillator output clock division factor. The output clock is divided by the factor OSCDIV+1.

Table 392. SXOSC_CTL register field descriptions (continued)

Field	Description
24 I_OSC	Crystal oscillator clock interrupt This bit can only be set by hardware when OSCCNT counter reaches the count value EOCV $\times$ 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred 1 Oscillator clock interrupt pending
30 S_OSC	Crystal oscillator status 0 Crystal oscillator output clock is not stable 1 Crystal oscillator is providing a stable clock
31 OSCON	Crystal oscillator power-down control 0 Crystal oscillator is switched off 1 Crystal oscillator is switched on

32.3 Functional description

The OSC32K circuit includes an internal oscillator driver and an external crystal circuitry. It can be used as a reference clock to specific modules depending on system needs.

The OSC32K can be controlled via the SXOSC_CTL register. The OSCON bit controls the power-down while bit S_OSC provides the oscillator clock available status.

After system reset, the oscillator is put to power-down state and software has to switch on when required. Whenever the OSC32K is switched on from off state, the OSCCNT counter starts and when it reaches the value EOCV[7:0] × 512, the oscillator clock is made available to the system. Also, an interrupt pending SXOSC_CTL[I_OSC] bit is set. An interrupt will be generated if the interrupt mask bit M_OSC is set.

The oscillator circuit can be bypassed by writing SXOSC_CTL[OSCBYP] bit to ‘1’. This bit can only be set by software. A system reset is needed to reset this bit. In this bypass mode, the output clock has the same polarity as the external clock applied on the OSC32K_EXTAL pin and the oscillator status is forced to ‘1’. The bypass configuration is independent of the power-down mode of the oscillator.

Table 393 shows the truth table of different configurations of the oscillator.

Table 393. OSC32K configurations

SXOSC_CTL fields		OSC32K_XTAL	OSC32K_EXTAL	OSC32K	Oscillator mode
OSCON	OSCBYP				
0	0	No crystal, high Z	No crystal, high Z	0	Power-down, IDDQ
x	1	x	External clock	OSC32K_EXTAL	Bypass, OSC Disabled
1	0	Crystal	crystal	OSC32K_EXTAL	Normal, OSC Enabled
		Ground	External clock	OSC32K_EXTAL	Normal, OSC Enabled

The OSC32K clock can be further divided by a configurable factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by SXOSC_CTL[OSCDIV] field.

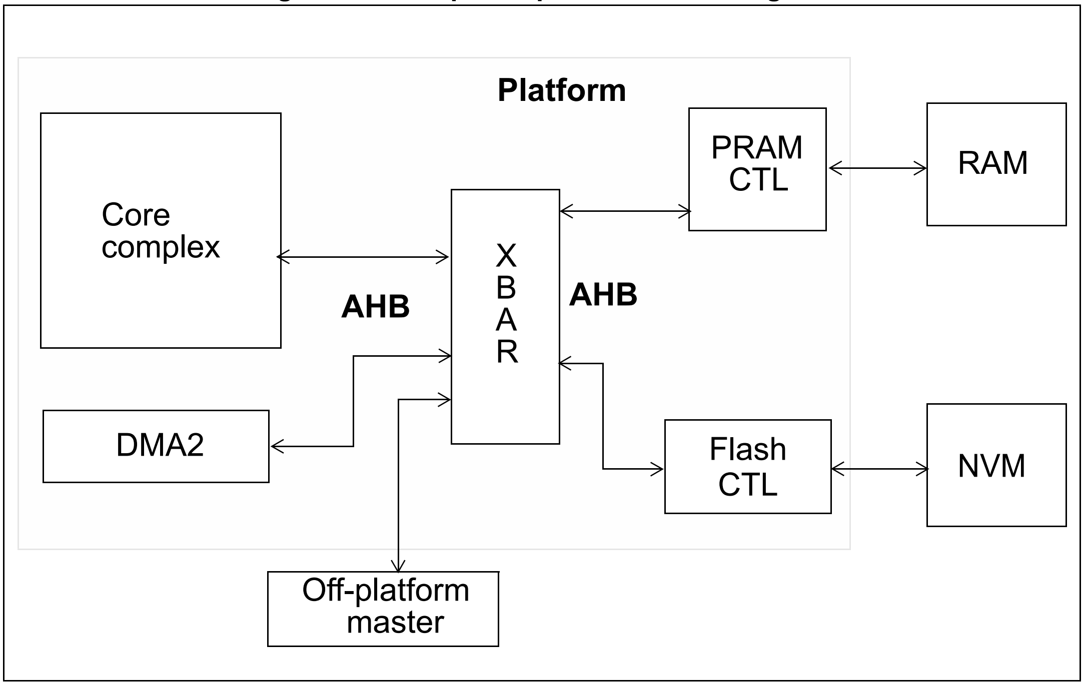
33 Platform RAM controller AHB (PRAMC_AHB)

33.1 Introduction

This section provides an overview of the Platform RAM Controller. The PRAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The PRAM controller supports two 64-bit AHB and one 64-bit RAM array interfaces. The first AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The second AHB port provides a connection from the flash controller to facilitate calibration overlay access. *Figure 303* is a simplified block diagram depicting the position of the PRAM controller within a typical platform architecture:

Figure 303. Simplified platform block diagram



The following list summarizes the key features of the PRAM controller:

- System bus supports 64-bit data and 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data and 8-bit ECC interface
- Late-write support via 64-bit data and 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than 64-bit data

The address path of the PRAM controller contains a multiplexer that chooses among the addresses presented on the system bus during read/write requests to RAM. These addresses are transferred to RAM array either immediately, for read accesses, or, for write accesses, through a temporary write address buffer or through a late-write address buffer.

The temporary write address buffer contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to simultaneously present the address and associated write data, because the write data is not valid until the AHB data phase. The late-write buffer holds write requests which are delayed to facilitate single-cycle responses on the system bus.

The read data is the result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire data path of the PRAM controller, including the late-write buffer, to support end-to-end ECC (e2eECC) coverage.

The write data path contains the multiplexer that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and stored directly in the RAM.

33.2 SRAM controller memory map and register definitions

The PRAM controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses will generate an IPS error.

Table 394. PRAMC_AHB memory map

Address offset (hex)	Register name	Reset value (hex)	Location
0x00	Platform RAM Configuration Register 1 (PRCR1)	0x0000_0200	*Section 33.2.1.1*

33.2.1 Registers description

33.2.1.1 Platform RAM configuration register 1 (PRCR1)

The Platform RAM configuration register 1 (PRCR1) is used to specify the operation of the PRAM controller.

Note: This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Offset: 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	PRI1	PRI0	P1_BO_DIS	P0_BO_DIS	0	0	0	0	0	FT_DIS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 304. Platform RAM configuration register 1 (PRCR1)

Table 395. PRCR1 field descriptions

Field	Description
22 PRI1	Not used, since port 1 is not used. This bit can be written by software.
23 PRI0	Not used, since port 1 is not used. This bit can be written by software.
24 P1_BO_DIS	Not used, since port 1 is not used. This bit can be written by software.
25 P0_BO_DIS	Port p0 read burst optimization disable 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response(1) when PRCR[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response(1) when PRCR[FT_DIS]=1
31 FT_DIS	Flow through disabled This field defines the AHB read response of the PRAM controller . The state of this field has no impact on the write response latency. This bit is cleared by hardware reset. **Note:** Do not change the FT_DIS bit value while accessing system RAM. Realocate code programming the FT_DIS bit to another area (for example, local core memory). 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

1. The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAM controller clock frequency. If system RAM is running at the same frequency as the PRAM Controller a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to PRAM Controller clock frequency.

33.3 Functional description

33.3.1 Read/Write introduction

The PRAM controller generates chip-select and write-enable, the array address, and write data as inputs to the RAM array. The PRAM controller captures read data from the RAM array and drives it onto the AHB along with the associated dataphase termination signals. To facilitate operating at higher frequencies, the PRAM controller can optionally incur a single wait state on the AHB response.

33.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the AHB.

33.3.1.2 Optional read wait-state

The PRAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCR1[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCR1[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCR1[FT_DIS] = 0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if PRCR1[FT_DIS] = 1, and a WRAP4 burst will have an access time of 3-2-2-2.

33.3.2 Writes

33.3.2.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back-to-back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Instead, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

33.3.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme’s 64-bit granularity. In the case of a read-modify-write action, the PRAM controller performs single-error correction, double-error detection on the read data. The write data is merged into the appropriate byte lanes along with the potentially-corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. For details on the ECC coding scheme, refer to *Section 33.5: Safety considerations*.

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of less than 64 bits require the insertion of one wait state before the data phase can be completed.

33.3.2.3 Unaligned writes

The PRAM controller is compliant with the AMBA-AHB2.v6 extensions specification concerning byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

Note: Unaligned writes always generate read-modify-write operations in the PRAM controller in order to preserve the validity of the ECC codeword.

33.4 Initialization/application information

Prior to reading, the ECC must be initialized by writing a known 64-bit value to each memory address. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Reading from an uninitialized address is likely to generate an uncorrectable ECC event.

The PRAM memory space can be initialized by storing a 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

33.5 Safety considerations

33.5.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named after M.Y. Hsiao, an IBM researcher who published a paper in the early 1970s about SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns
2. Every column is distinct
3. Every column contains an odd number of ones, and hence is “odd weight”

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field.

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in *Table 396*. A ‘*’ in *Table 396* indicates the corresponding data or address bit is XOR’ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR’ed) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses the AHB bit numbering convention where bit[0] is the least significant bit.

Table 396. PRAM controller basic H-matrix definition

Checkbits [7:0]	Data Bit[(1)]																															
	Byte 7								Byte 6								Byte 5								Byte 4							
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
7	*				*					*			*		*	*		*	*	*	*					*	*			*	*	
6			*	*			*	*					*		*		*				*				*			*	*			*
5	*	*	*	*					*				*	*		*		*	*				*	*	*				*			
4	*		*			*		*					*		*		*			*			*	*			*				*	
3									*	*	*	*	*		*				*		*				*	*	*	*				
2					*		*					*		*			*	*				*	*							*		*
1		*		*		*		*	*	*	*	*						*		*		*		*						*		*
0		*			*	*	*				*			*	*	*		*				*				*		*	*		*	

Checkbits [7:0]	Byte 3								Byte 2								Byte 1								Byte 0							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7					*		*				*		*				*	*			*	*			*				*	*		*
6	*	*		*	*				*				*				*	*		*	*					*	*				*	*
5			*	*			*	*	*			*	*	*					*		*	*		*								
4	*		*	*				*		*	*				*	*	*			*			*	*				*		*		
3	*	*	*			*						*	*		*	*			*				*		*	*	*			*		
2		*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
1		*						*		*			*	*	*			*				*				*			*	*	*	
0		*		*		*		*				*				*				*				*				*	*		*	*

Checkbits [7:0]	Address Bit[(1)]																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7		*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*				
6		*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
5		*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*			
4	*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*				
3	*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*			
2	*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*			
1	*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*			
0	*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*			

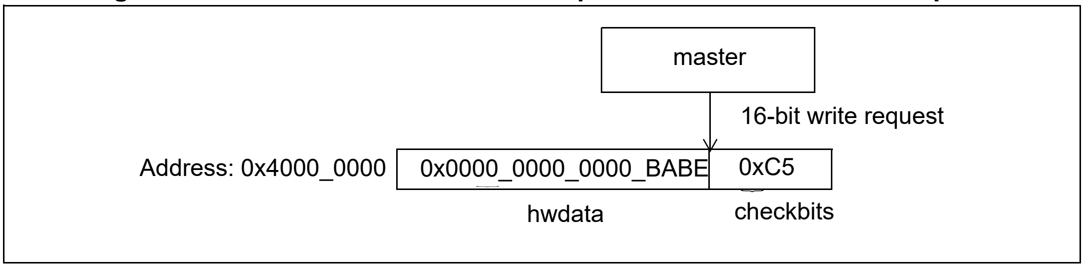
1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

33.5.1.1 E2EECC considerations on less-than-64-bit write transactions

Writes of less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme 64-bit granularity. In the case of a read-modify-write action, the PRAM controller performs SEC/DED on the read data. The PRAM controller uses the following checkbit manipulation technique to ensure that any faults in the PRAM control logic, associated with performing the read-modify-write action, will ultimately be detectable by e2eECC:

- On an 8-bit, 16-bit or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non pertinent byte lanes zeroed out, and the associated checkbit is based on this “zero-padded” write data.
  Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

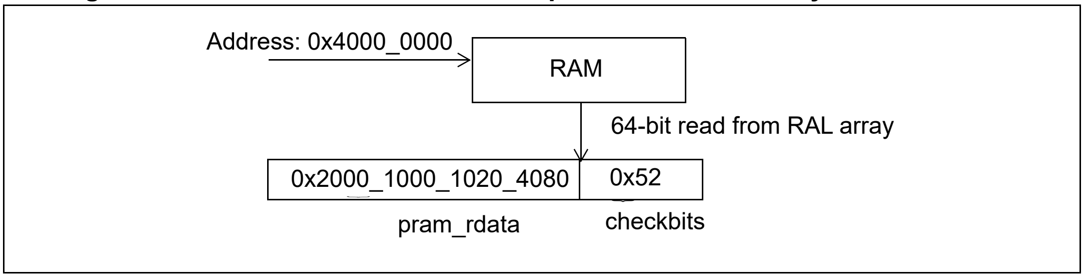
Figure 305. E2EECC consideration for partial write - 16-bit write request



- The PRAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

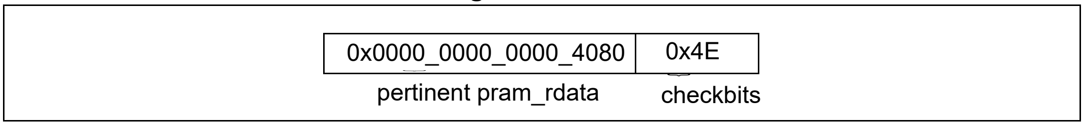
  If a non correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.

Figure 306. E2EECC consideration for partial write - memory read from RAM



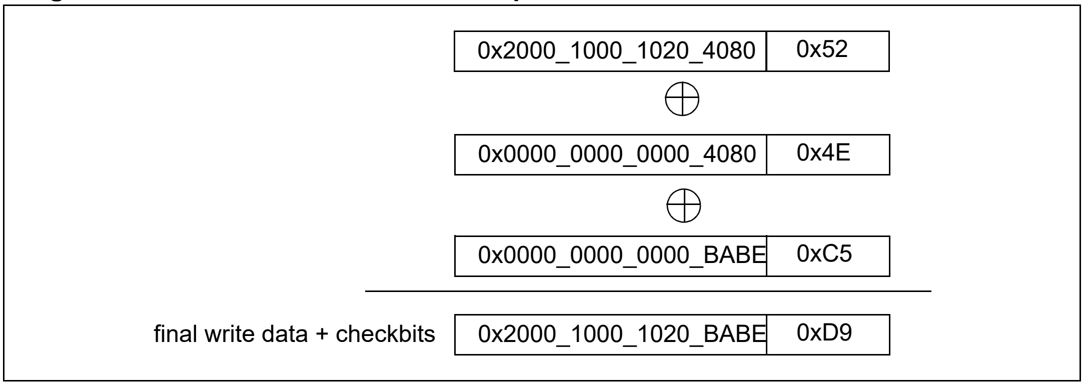
- The PRAM controller isolates the pertinent byte lanes of the read data by zero-padding the non pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.

Figure 307. E2EECC consideration for partial write - Separation of check-bits of targeted bit-lines



- The PRAM controller merges the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.

Figure 308. E2EECC consideration for partial write - calculation of final check-bits



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and PRAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The PRAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

33.5.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by an e2eECC check performed by the requesting master. Functional safety coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the PRAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions

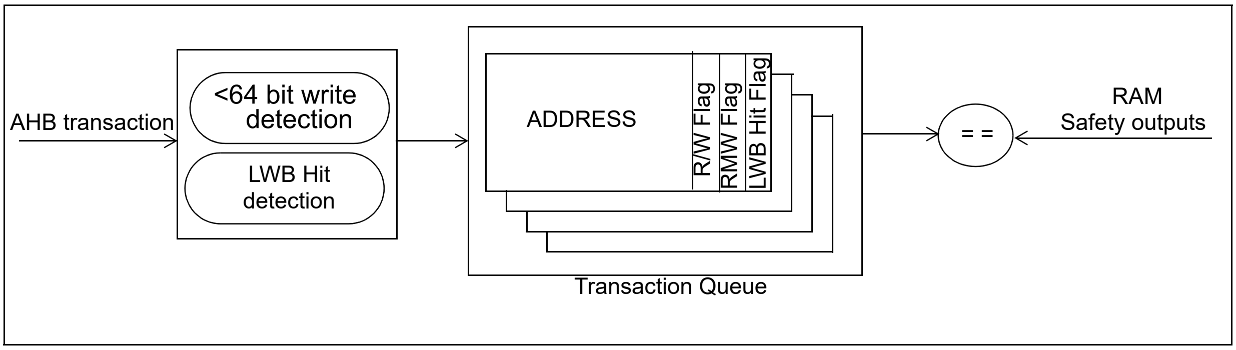
- Direct system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, the safety feature compare outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

Figure 309. RAM controller transaction monitor block diagram



34 Flash memory controller

34.1 Introduction

The Flash memory controller act as an interface between the system bus (AHB-Lite 2.v6) and the Flash memory arrays.

The Flash memory controllers support two 64-bit AHB buses and a 256-bit read data interface from each Flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the Flash arrays into the mini-cache. This buffer mechanism serves to deliver Flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed Flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the Flash array page size and providing 256 bytes of total buffered storage.

Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the Flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage.

34.2 Features

The following list summarizes the key features of the Flash memory controller:

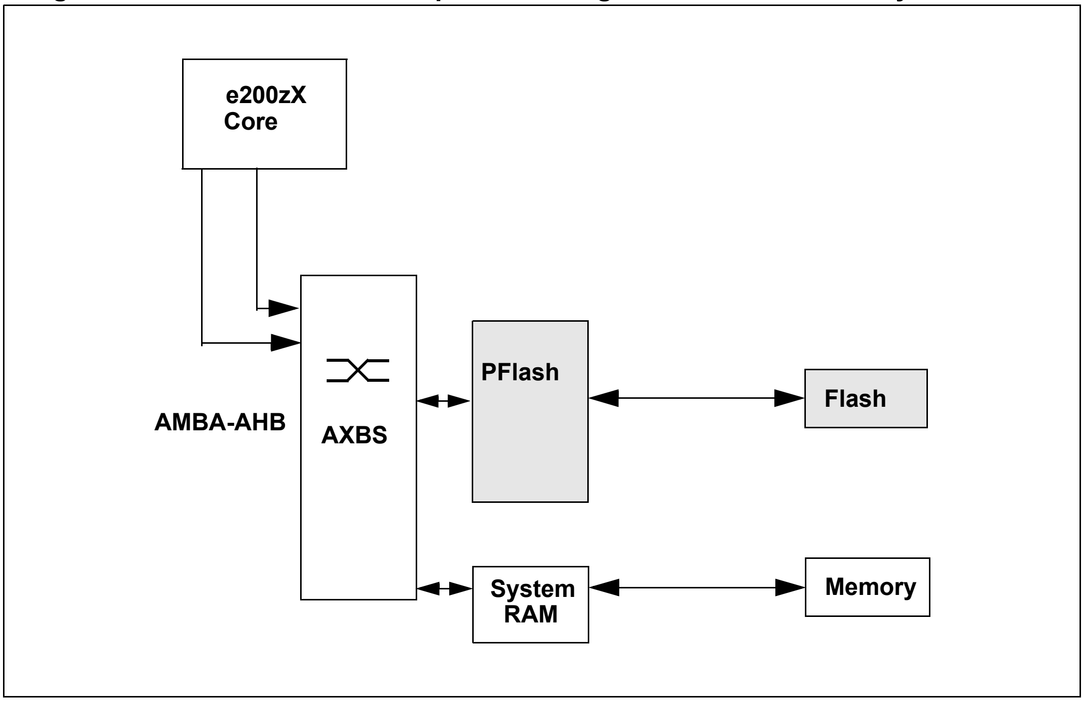
- Two 64-bit AHB interface ports (p0, p1) allowing simultaneous access to dedicated prefetch mini-cache per slave port
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle “buffer hit” read response
- Configurable access control based on read/write and AHB master ID attributes
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets
- Optional address pipelining capability to maximize throughput
- Support for reporting of single-, double- and multi-bit Flash ECC events on a 64-bit doubleword boundary

34.3 Block diagrams

Figure 310 provides a block diagram showing the Flash memory controller and the attached Flash array.

Note: This module is also known as the Platform Flash controller, that is, PFlash.

Figure 310. Platform-centric simple block diagram with Flash memory controllers



34.4 Flash memory controller memory map

34.4.1 Overview

The Flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of Flash memory access configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The Flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the Flash memory controller module is in the midst of an operation will result in non-deterministic behavior. Software must be architected to avoid this scenario.

There is no idle indicator for the Flash controller. The recommended flow for multi-core devices is to start only one core and execute initialization code to completion before starting the remaining cores. If the user needs to reconfigure the Flash, code execution must be temporarily moved to system RAM.

Table 397. Flash memory controller memory map

Address Offset	Register	Section
00x00	Platform Flash configuration register 1 (PFCR1)	*Section 34.4.1.1*
0x004	Platform Flash configuration register 2 (PFCR2)	*Section 34.4.1.2*
0x008	Platform Flash configuration register 3 (PFCR3)	*Section 34.4.1.3*
0x00c	Platform Flash access protection register (PFAPR)	*Section 34.4.1.4*

34.4.1.1 Platform Flash configuration register 1 (PFCR1)

The PFlash configuration register 1 (PFCR1) controls operation Port p0 of the Flash memory controller.

Offset 0x000 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC			RWSC					0	P0_DPFEN	0	P0_IPFEN	0	P0_PFLIM		P0_BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

Figure 311. Platform Flash configuration register 1 (PFCR1)

Note: See the chip-specific Flash controller information for details about the actual masters available on the device.

Table 398. PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 398. PFCR1 field descriptions (continued)

Field	Description
2 P0_M13PFE	Port0 Master 13 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 398. PFCR1 field descriptions (continued)

Field	Description
11 P0_M4PFE	Port0 Master 4 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16:18 APC	Address Pipeline Control This field controls the number of cycles that a subsequent Flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1. 000 Pipelined access to the Flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 Reserved 011 Reserved 1xx Pipelined access to the Flash is disabled and one wait state is inserted before a subsequent access can be initiated

Table 398. PFCR1 field descriptions (continued)

Field	Description
19:23 RWSC	Read Wait State Control This field controls the number of wait-states to be added to the best-case Flash array access time for reads. The best-case Flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the Flash memory controller and the actual read access time of the Flash memory controller. The required settings are documented in the device data sheet. Higher operating frequencies require non-zero settings for this field for proper Flash operation. This field applies to the configuration of Port0 and Port1. 00000 No additional wait-states are added 00001 One additional wait-state is added ... 11111 Thirty-one additional wait-states are added **Note:** Refer to the device data sheet for wait state configuration
25 P0_DPFEN	Port0 Data Prefetch Enable This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
27 P0_IPFEN	Port0 Instruction Prefetch Enable This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
29:30 P0_PFLIM	Port0 PFlash Prefetch Limit This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 1x The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

34.4.1.2 Platform Flash configuration register 2 (PFCR2)

The PFlash configuration register 2 (PFCR2) specifies operation Port1 of the Flash memory controller.

Offset 0x004 Access: Supervisor Read/Write

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	P1_M15PFE	P1_M14PFE	P1_M13PFE	P1_M12PFE	P1_M11PFE	P1_M10PFE	P1_M9PFE	P1_M8PFE	P1_M7PFE	P1_M6PFE	P1_M5PFE	P1_M4PFE	P1_M3PFE	P1_M2PFE	P1_M1PFE	P1_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	P1_DPFEN	0	P1_IPFEN	0	P1_PFLIM		P1_BFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 312. Platform Flash configuration register 2 (PFCR2)

Note: See the chip-specific Flash controller information for details about the actual masters available on the device.

Table 399. PFCR2 field descriptions

Name	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P1_M11PFE	Port1 Master 11 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 399. PFCR2 field descriptions (continued)

Name	Description
5 P1_M10PFE	Port1 Master 10 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P1_M5PFE	Port1 Master 5 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P1_M4PFE	Port1 Master 4 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P1_M2PFE	Port1 Master 2 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 399. PFCR2 field descriptions (continued)

Name	Description
14 P1_M1PFE	Port1 Master 1 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
25 P1_DPFEN	Port1 Data Prefetch Enable This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
27 P1_IPFEN	Port1 Instruction Prefetch Enable This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
29:30 P1_PFLIM	Port1 PFlash Prefetch Limit This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 1x The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

34.4.1.3 Platform Flash configuration register 3 (PFCR3)

Offset 0x008 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_WCFG		P1_WCFG		0	0	0	0	0	0	0	0	0	0	0	BAF_DIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ARBM		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 313. Platform Flash configuration register 3 (PFCR3)

Table 400. PFCR3 field descriptions

Name	Description
0:1 P0_WCFG	Port0 Way Configuration This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the Flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the Flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers in an indexed set are available for any Flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers in an indexed set are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved.
2:3 P1_WCFG	Port1 Line Buffer Configuration This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the Flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the Flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any Flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with buffer 0 allocated for instruction fetches and buffer 1 for data accesses. 11 Reserved

Table 400. PFCR3 field descriptions (continued)

Name	Description
15 BAF_DIS	BAF Disable This field controls executable access to the BAF (Boot Assist Flash) region of the Flash. Once this field is set, attempted instruction accesses targeting the BAF region are aborted and terminated with a system bus error. The affect of this field applies to system bus transfers through both, Port0 and Port1. Data-type accesses to the BAF region are not affected by this field. Once this field is set, it becomes a read-only field and can only be cleared by hardware reset. Once this field is set, any subsequent write attempts to modify this field are ignored with an error-free data transfer termination. This field is initialized by hardware reset. 0 Executable access to the BAF Flash region is allowed. 1 Executable access to the BAF Flash region is prohibited.
16:17 ARBM	Arbitration Mode This 2-bit field controls the arbitration of concurrent Flash access requests from the two AHB ports of the Flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever both ports issue concurrent requests. This field is initialized by hardware reset. 00 Fixed priority arbitration with AHB p0>p1 01 Fixed priority arbitration with AHB p1>p0 1x Round-robin arbitration

34.4.1.4 Platform Flash access protection register (PFAPR)

The PFlash access protection register (PFAPR) is used to control read and write accesses to the Flash array.

Offset 0x00C Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 314. Platform Flash access protection register (PFAPR)

Note: See the chip-specific Flash controller information for details about the actual masters available on the device.

Table 401. PFAPR field descriptions

Field	Description
0:1 M0AP	Master 0 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2:3 M1AP	Master 1 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4:5 M2AP	Master 2 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
6:7 M3AP	Master 3 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8:9 M4AP	Master 4 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10:11 M5AP	Master 5 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 401. PFAPR field descriptions (continued)

Field	Description
12:13 M6AP	Master 6 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
14:15 M7AP	Master 7 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16:17 M8AP	Master 8 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
18:19 M9AP	Master 9 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20:21 M10AP	Master 10 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22:23 M11AP	Master 11 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 401. PFAPR field descriptions (continued)

Field	Description
24:25 M12AP	Master 12 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26:27 M13AP	Master 13 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
28:29 M14AP	Master 14 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30:31 M15AP	Master 15 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

34.5 Functional description

As shown in *Figure 310* the Flash memory controllers interface between:

- The AHB system bus ports
- Flash memory array

For accesses targeting the Flash array, the Flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the Flash array. The Flash memory controller captures read data from the Flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of two ways of the Flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

34.5.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

34.5.2 Access protections

34.5.2.1 PFAPR - Platform Flash access protection register (PFAPR)

The Flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash access protection register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in *Section 34.4.1.4*. Detection of a protection violation results in an error response from the Flash memory controller on the AHB transfer.

34.5.2.2 BAF execution disable

The Flash memory controller provides a mechanism to disable instruction execution of the BAF (Boot Assist Flash) code. Setting the BAF Disable field (BAF_DIS) in the PFlash configuration register 3 (PFCR3) prevents execution of the BAF instructions. When this field is set, all incoming requests from the system bus on either Port0 or Port1 are evaluated. If the Flash memory controller detects an attempted instruction fetch access to the BAF region of the Flash, the Flash memory controller aborts the access and returns a system bus error. Data accesses to the BAF region are not affected by the state of PFCR3[BAF_DIS].

34.5.3 Read cycles – buffer miss

On an incoming AHB read request, a mini-cache lookup and access privilege evaluation are performed during the AHB address phase. In the event of a buffer miss, a Flash access is initiated.

If the Flash access was the direct result of an AHB transaction, the corresponding page buffer is loaded and marked as most-recently-used. If the Flash access was the result of a speculative prefetch to the next sequential line, it is loaded into the least-recently-used buffer. The status of this buffer is not changed to most-recently-used until a subsequent buffer hit occurs as a result of an AHB read request.

34.5.4 Read cycles – buffer hit

Single cycle read responses to the AHB are possible with the Flash memory controller when the requested read access was previously loaded into one of the page buffers. In these “buffer hit” cases, read data is returned on the system bus with a zero wait-state response.

34.5.5 Error termination

The Flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding Read access Control or Write access Control settings do not allow the access, thus causing a protection violation. See *Section 34.4.1.4* for more detail. In this case the Flash memory controller does not initiate a Flash array access.
- Attempted instruction fetch by an AHB master to the BAF Flash region when the PFCR3[BAF_DISABLE] field in Platform Flash configuration register 3 (PFCR3) is set. The assertion of PFCR3[BAF_DISABLE] indicates that code stored in the BAF region should not be executed. See *Section 34.4.1.3* for more detail. In this case, the Flash memory controller does not initiate a Flash array access.
- The Flash returns an error response on an attempted access (see *Section 34.5.6*).
- Attempted access by an AHB master to a reserved region in the Flash memory map.

34.5.6 Flash error response operation

The Flash array may signal an error response to terminate a requested access due to improper sequencing during program/erase operations, or improper sequencing during array integrity testing. When an error response is received, the Flash memory controller does not update or validate a page read buffer. An error response may be signaled on read or interlock write operation. For more information on the specifics related to signaling of Flash errors, including Flash ECC events, array integrity testing and read-while-write events, refer to the Flash memory chapter.

34.5.7 Security module exclusive control

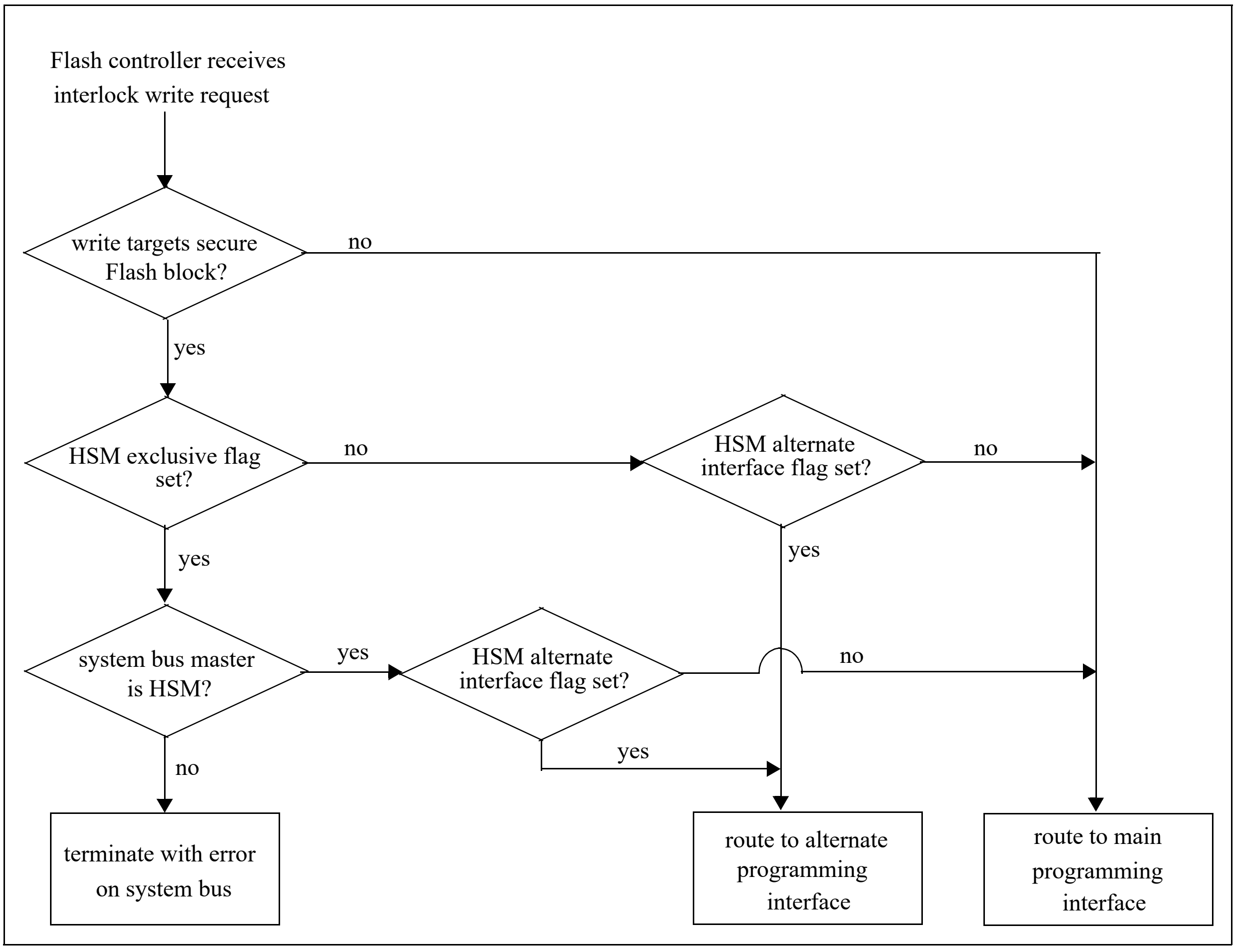
During “Production at OEM” lifecycle state and beyond, the HSM can establish the secure Flash blocks to be exclusive to the HSM and route program or erasing of secure Flash blocks to the alternate programming interface.

The secure Flash space is independently controlled for each of the following regions:

- 16 KB secure code Flash block at address range 0x0060_C0000 - 0x0060_FFFF
- 128 KB secure code Flash block at address range 0x0061_0000 - 0x0062_FFFF
- Two 16 KB secure data Flash blocks at address range 0x0068_0000 - 0x0068_7FFF

The HSM provides flags to individually control the HSM-exclusivity for each of the four secure Flash regions. When a given secure Flash block is marked as exclusive to the HSM, the HSM is the only system bus master granted access to that block and access attempts by non-HSM masters are terminated with error. Once HSM-exclusivity is established for a given secure Flash block, the HSM is guaranteed access to that block, regardless of the state of the censorship controls. Secure Flash exclusivity applies to read and write accesses.

Figure 315. Interlock Write Alternate Programming Interface Routing Flowchart



34.5.8 Access pipelining

Accesses to the Flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the Flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by the AHB master. Access pipelining may be applied only to read cycles targeting the Flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal Flash access.

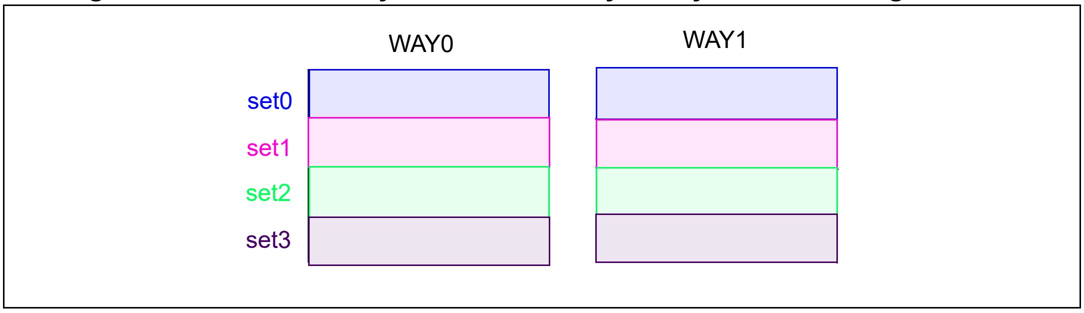
Note: Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.

34.5.9 Line read buffers and prefetch operation

The AHB ports of the Flash memory controller each contain a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the Flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see Section 34.4.1.1). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the Flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32 bytes, and a subsequent Flash access is initiated. A Flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

Figure 316. Flash memory controller 4-entry, 2-way mini-cache organization



Once the candidate line buffer has been selected, the Flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of Flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted Flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in Section 34.4.1.1. More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR{1,2}[P{0,1}_BFEN] must be set to ‘1’,
2. PFCR{1,2}[P{0,1}_PFLIM] must be non-zero, and
3. either PFCR{1,2}[P{0,1}_IPFEN] or PFCR{1,2}[P{0,1}_DPFEN] must be asserted

Refer to Section 34.4.1.1 for a description of these controls.

34.5.10 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to Section 34.4.1.1 for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to

Section 34.4.1.2 for a description of these controls. Prefetches are never triggered by write cycles.

34.5.11 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to Section 34.4.1.1 for a description of these controls.

34.5.12 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1}_WCFG) field. Refer to Section 34.4.1.3. The buffers can be organized as a “pool” of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, way 0 is allocated for instruction fetches and way 1 for data accesses.

34.5.13 Safety considerations

34.5.13.1 e2eECC (End-to-End ECC)

34.5.13.1.1 e2eECC and Flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with Flash memory. The basic issues involve: the default state of a Flash block that is erased (all ones) and the fact that programming an erased Flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the Flash memory in three ways.

First, the all ones codeword, since it reflects the state of an erased Flash location, is treated as a valid, error free encoding; in particular, it is required that the checkbits associated with the all ones data value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased Flash block generates an all ones data for all locations, the address field cannot be included in the ECC code used by the Flash. The Flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the Flash memory and its support for EEPROM emulation.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the Flash memory in multiple ways. The required ECC adjustments are handled by the platform Flash controller before data is written to the Flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a Flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all ones erased state.

On a Flash read operation, a similar but “reversed” set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the Flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following Flash ECC example. Let the Flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns fl_rdata = 0xFFFFFFFF_FFFFFFFF, fl_cdata = 0xFF. The Flash memory controller calculates the addr_chkbit = 0xC4 using the H-matrix; it inverts the fl_cdata vector to 0x00 and factors in the addr_chkbit = 0xC4 to produce the following valid e2eECC codeword:

```
addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)
```

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

```
addr = 0x00345678, wdata = 0xBABEFACE_DEADBEEF, wchkbit = 0xE3
```

Before performing the interlock write to the Flash array, the Flash memory controller adjusts the chkbit value as fl_wchkbit = 0xD8 (0xE3 ^ 0xff ^ 0xC4) by toggling the original write checkbits and then factoring in the addr_ecc. Accordingly, the codeword stored in the Flash array is:

```
fl_data = 0xBABEFACE_DEADBEEF, fl_checkbit = 0xD8
```

3. Read address 0x00345678. The Flash array returns the stored codeword, fl_rdata = 0xBABEFACE_DEADBEEF, fl_rchkbit = 0xD8. The Flash memory controller inverts the checkbit vector and factors in the address checkbits (0xD8 ^ 0xFF ^ 0xC4) to produce:

```
addr = 0x00345678, rdata = 0xBABEFACE_DEADBEEF, rchkbit = 0xE3
```

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The Flash memory controller performs EDC after ECC check on all Flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

34.5.13.2 Flash address generation check

Functional safety coverage of the address and data are handled by ECC performed within the Flash and e2eECC performed at the master. Functional safety coverage of the address path and control within the Flash memory controller rely on a feedback path between the Flash controller and Flash. Recall on a requested access to Flash, the Flash memory controller must decode the system AHB bus signals to generate the corresponding Flash interface signals to invoke a Flash lookup. In addition to providing the requested read data, the Flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.In the event of EDC after ECC errors detected on accesses to data and secure data flash, the event is not reported.

This sideband information is used by the Flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the Flash memory controller or the transmission path between the Flash memory controller and Flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

34.5.14 e2eECC on data Flash accesses

ECC events detected on accesses to the EEPROM Flash blocks are suppressed from being reported to the Memory Error Management Unit (MEMU).

In the event of a single-bit or double-bit correction, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is suppressed from being reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode value is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address, and the non-correctable error event is suppressed from being reported to the MEMU. In the event of EDC after ECC errors detected on accesses to data and secure data flash, generation of the fault is suppressed. The related bit in MCR is not set and the event is not forwarded to the FCCU.

34.5.15 Array integrity considerations

During an array integrity sequence, the Flash memory array ignores any incoming read requests. When a Flash array integrity check is in progress, the Flash memory controller terminates all Flash access requests with an error.

35 Embedded Flash Memory

35.1 Introduction

The Flash Memory module serves as electrically programmable and erasable nonvolatile memory, for instructions and data storage.

The Flash Memory module is a nonvolatile, solid-state silicon memory device consisting of blocks of single transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements.

The Flash Memory module has two functional units:

- The Flash core: it is composed of a common high voltage HV block and a Flash array which can contain one or more Flash partitions. Each Flash partition is composed of arrayed nonvolatile storage elements, sense amplifiers, row decoders and column decoders. The arrayed storage elements in the Flash partitions are subdivided into physically separate units referred to as blocks.
- The Memory interface: it contains the registers and logic which control the operation of the Flash core and contains two main ports: the Array Interface and the Registers Interface. The memory interface is also the interface between the Flash Memory module and a Bus Interface Unit (BIU) and contains the ECC and redundancy logic.

35.1.1 Overview

The 2320 KB Flash Memory module contains two array partitions. Read-While-Write operations are only possible for separate partitions, meaning fetching/reading from one (or more) partition(s) while a program/erase operation is active on another partition. Each module counts as an RWW partition.

EEPROM emulation is managed as 4 × 16 KB blocks in one partitions, referred to as Data Flash.

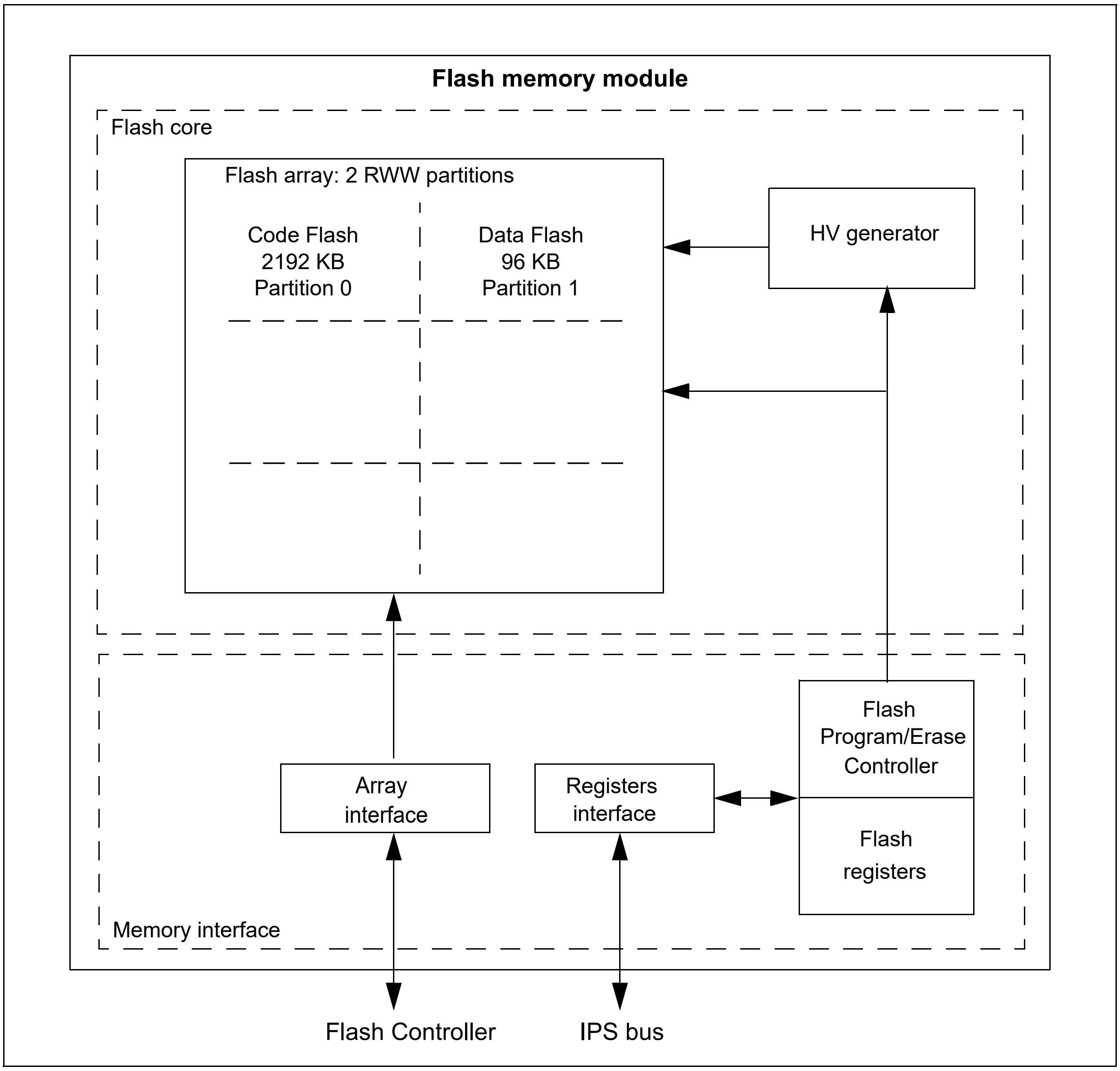
HSM EEPROM emulation is managed as 2 × 16 KB blocks in one partition, referred to as HSM Data Flash.

The Modify operations are managed by an embedded Flash Program/Erase Controller (FPEC). Commands to the FPEC are delivered through a User Registers interface.

The Read Data bus is 256-wide, while the Flash registers are on a separate, 32-bit bus.

The high voltages needed for Program and Erase operations are internally generated and are common for the two partitions.

Figure 317. Flash memory module structure



The Flash Memory module is designed for use in embedded applications which require high density non-volatile memories with high-speed Read access.

The Flash array is addressable:

- For programs in data partition: by word (32 bits) or doubleword (64 bits). ECC granularity is 64 bits.
- For programs in code partition: by word (32 bits) or doubleword (64 bits). ECC granularity is 128 bits.
- For reading code partitions – by page (256 bits).
- For reading data partitions – by half-page (128 bits).

Multiple-word or doubleword writes to the Flash memory may be performed to fill the program page buffer (256 bits), enabling page programming (256 bits – requiring 4 doubleword writes) and quad-page programming (1024 bits – requiring 16 doubleword writes). Code Flash memory reads always return 256, although Read buffering may be performed by the Bus Interface Unit (BIU). The address for each word retrieved within a page differs from the other addresses in the page by address bits (4:2).

Data Flash memory reads always return 128 bits, or two consecutive doublewords of information. The address for each word retrieved within a doubleword differs from the other addresses by address bits (3:2).

The Flash Memory module supports fault tolerance through Error Correction Code (ECC) and error detection. The ECC implemented within the Data Flash Memory module corrects up to double bit failures and detects triple bit failures (as uncorrectable).The ECC implemented within the Code Flash Memory module corrects double bit failures and detects triple bit failures (as uncorrectable).

The Flash Memory module uses an embedded hardware algorithm implemented in the Flash Program/Erase Controller to program and erase the Flash partitions.

Control logic that works with the software block-enables and software lock mechanisms are included in the embedded hardware algorithm to guard against accidental program/erase. The hardware algorithm performs the necessary steps to ensure that the storage elements are programmed and erased with sufficient margin to guarantee data integrity and reliability.

A programmed bit in the Flash array reads as logic level ‘0’ (or low). An erased bit in the Flash array reads as logic level ‘1’ (or high).

Programming and Erasing of the Flash array requires multiple system clock cycles to complete.

The Program and Erase sequences may be suspended or aborted.

35.1.2 Features

- 30 ns code/data array access time
- Code Partition High Read parallelism: 256 bits
- Data Partition Read parallelism: 128 bits
- Error Correction Code (Double Error Correction-Triple Error Detection) to enhance data integrity
- Data Partition Doubleword program granularity: 64 bits
- Erase with fast erase time at factory
- Blocks one time programmable (OTP) marking
- Read-While-Modify capability when the accesses are to different partitions
- Erase/Program Suspend
- Programming during Erase-Suspend
- Software programmable program/erase protection to avoid unwanted writings
- Temperature range (-40 °C to 150 junction)
- UTEST mode (user-accessible Test modes)
- Alternate interface provided for program and erase operations to dedicated master as private access to blocks
- Critical flops, which may affect Flash content, are triple-voted for safety application.

35.1.3 Modes of operation

The Flash Memory module supports the following modes of operation:

- Reset: the Flash Memory module initializes when the reset is deasserted.
- Power-down (Disable mode): the Flash Memory module removes all static consumption. The embedded Flash memory (array and registers) is not accessible to Read, Program or Erase once disabled. The wake-up time from Disable mode is longer than the wake-up time from Sleep mode.
- Low-power mode (Sleep mode): the Flash Memory module removes most of the static consumption. It is not accessible to Read, Program or Erase once disabled, however it does maintain a faster exit to Read mode with respect to Power-Down.
- Read mode: the Flash Memory module accepts Read accesses to the array.
- Modify mode: it is possible to read and write registers, Interlock Write the memory array, program the memory array, and erase the memory array. Program and Erase operations are initiated by performing array and register writes, and controlled by an internal state machine. Program/Erase operations or User Test modes (Margin Read or Array Integrity Check) are considered Main Modify modes.
- Read-While-Write: The Flash Memory module accepts Read accesses to array partitions that are not in Modify mode.
- Alternate Modify mode: blocks in LOW and MID space could be assigned to Alternate interface, though which it is possible to read and write registers, Interlock Write the assigned memory array, program the assigned memory array, and erase the assigned memory array. Program and Erase operations are initiated by performing array and alternate register writes, and are controlled by the same internal state machine as in Main. Alternate Program/Erase operations are considered Alternate Modify modes. Alternate Modify modes operates concurrently to Main ones and, although functionally independent, timing can be mutually influenced.

Such modes of operation are described in detail in *Section 35.4: Functional description*.

35.2 Flash memory map and description

The Flash Memory module block subdivision also facilitates independent Erase and Program protection. A software mechanism is provided to independently lock and unlock each block in Low, Mid, High and Data address spaces against program and erase.

35.2.1 Flash array memory map

35.2.1.1 Code Flash

Partition 0 accounts for 2192 KB of the total Flash Memory and is divided in 14 user blocks for code storage.

The sectorization of partition 0 of the Flash Memory is shown in *Table 402*.

Table 402. Flash memory partition 0 memory map

Block	Addresses	Size	Address space	LOCK	SEL
B0F0	0x00FC_0000 to 0x00FC_3FFF	16 KB	Low	LOWLOCK[1]	LOWSEL[1]
B0F1	0x00FC_4000 to 0x00FC_7FFF	16 KB	Low	LOWLOCK[2]	LOWSEL[2]
B0F2	0x00FC_8000 to 0x00FC_FFFF	32 KB	Low	LOWLOCK[4]	LOWSEL[4]
B0F3	0x00FD_0000 to 0x00FD_7FFF	32 KB	Low	LOWLOCK[5]	LOWSEL[5]
B0F4	0x00FD_8000 to 0x00FD_FFFF	32 KB	Low	LOWLOCK[6]	LOWSEL[6]
B0F5	0x00FE_0000 to 0x00FF_FFFF	128 KB	256K	A256KLOCK[0]	A256KSEL[0]
B0F6	0x0100_0000 to 0x0103_FFFF	256 KB	256K	A256KLOCK[1]	A256KSEL[1]
B0F7	0x0104_0000 to 0x0107_FFFF	256 KB	256K	A256KLOCK[2]	A256KSEL[2]
B0F8	0x0108_0000 to 0x010B_FFFF	256 KB	256K	A256KLOCK[3]	A256KSEL[3]
B0F9	0x010C_0000 to 0x010F_FFFF	256 KB	256K	A256KLOCK[4]	A256KSEL[4]
B0F10	0x0110_0000 to 0x0113_FFFF	256 KB	256K	A256KLOCK[5]	A256KSEL[5]
B0F11	0x0114_0000 to 0x0117_FFFF	256 KB	256K	A256KLOCK[6]	A256KSEL[6]
B0F12	0x0118_0000 to 0x011B_FFFF	256 KB	256K	A256KLOCK[7]	A256KSEL[7]
B0F13	0x0060_C000 to 0x0060_FFFF	16 KB	Low	LOWLOCK[3]	LOWSEL[3]
B0F14	0x0040_0000 to 0x0040_3FFF	128 KB	Low	LOWLOCK[7]	LOWSEL[7]
B0UT	0x0040_0000 to 0x0040_3FFF	16 KB	Low	TSLOCK	—
B0BF	0x0040_4000 to 0x0040_7FFF	16 KB	Low	LOWLOCK[0]	—

35.2.1.2 Data Flash

Data Flash is allocated in partition 1 of the Flash Memory as 4 × 16 KB blocks destined for data storage.

The Data Flash sectorization of the Flash Memory module is listed below.

Table 403. Flash Memory EEPROM DATA Partition 1 memory map

Block	Addresses	Size	Address Space	LOCK	SEL
D0_00	0x0080_0000 to 0x0080_3FFF	16 KB	High	HIGHLOCK[0]	HIGHSEL[0]
D0_01	0x0080_4000 to 0x0080_7FFF	16 KB	High	HIGHLOCK[1]	HIGHSEL[1]
D0_02	0x0080_8000 to 0x0080_BFFF	16 KB	High	HIGHLOCK[2]	HIGHSEL[2]
D0_03	0x0080_C000 to 0x0080_FFFF	16 KB	High	HIGHLOCK[3]	HIGHSEL[3]

HSM Data Flash is allocated in partition 1 and is composed by 2 x 16 KB blocks. The HSM Data Flash sectorization is listed below.

Table 404. Flash Memory HSM EEPROM Data Partition 1 memory map

Block	Addresses	Size	Address Space	LOCK	SEL
H0_00	0x0068_0000 to 0x0068_3FFF	16 KB	Mid	MIDLOCK[0]	MIDSEL[0]
H0_01	0x0068_4000 to 0x0068_7FFF	16 KB	Mid	MIDLOCK[1]	MIDSEL[1]

35.2.1.3 UTEST block memory map

The UTEST block is programmed and read independently of the other blocks. UTEST is included to support systems which require nonvolatile memory for security and to store system initialization information.

The usage of reserved UTEST block is detailed in *Table 405*.

Table 405. UTEST block memory map

Address Offset	Description	Size	Access
UTEST (Base Address 0x0040_0000 to 0x0040_3FFF)			
0x0000	Reserved	32 bytes	R
0x0020	Factory Erase Diary Location	16 bytes	R/P
0x0030	Test Mode Disable Override Passcode	16 bytes	R/P
0x0040	Test Mode Disable Seal	16 bytes	R/P
0x0050	Test Mode Disable Group A	16 bytes	R/P
0x0060	Test Mode Disable Group B	16 bytes	R/P
0x00A0	UID	32 bytes	R
0x00C0	SW-DCF Start address	16 bytes	R
0x00D0	Reserved	48 bytes	R
0x0100	Password to enable the Flash Test Mode when in FA	32 bytes	R
0x0120	JTAG Password	32 bytes	R/P
0x0140(1)	PASS password - Group 0	32 bytes	R/P
0x0160	PASS password - Group 1	32 bytes	R/P
0x0180	PASS password - Group 2	32 bytes	R/P
0x01A0	PASS password - Group 3	32 bytes	R/P
0x01C0	Reserved	32 bytes	R/P
0x01E0	Life Cycle slot 0 - ST Production	32 bytes	R/P
0x0200	Life Cycle slot 1 - Customer Delivery	32 bytes	R/P
0x0220	Life Cycle slot 2 - OEM Production	32 bytes	R/P
0x0240	Life Cycle slot 3 - In Field	32 bytes	R/P
0x0260	Life Cycle slot 4 - Failure Analysis	32 bytes	R/P
0x0280	Customer Single Bit Correction area	32 bytes	R
0x02A0	Customer Double Bit Correction area	32 bytes	R

Table 405. UTEST block memory map (continued)

Address Offset	Description	Size	Access
0x02C0	Customer Triple Bit Detection area	32 bytes	R
0x02E0	Customer Programmable Detection Area	32 bytes	R
0x0300	DCF Records	3328 bytes	R/P
0x1000	Customer OTP data	12288 bytes	R/P

1. The offset 0x140 contains the least significant 32-bit of the PASS password - Group 0. The offset 0x15C contains the most significant 32-bit of the PASS password - Group 0. The same endianess also applies to the other groups of the PASS password.

UTEST block supports RWW, and is grouped with the Code Flash in partition 0.

Programming of any section of the UTEST block is subject to similar restrictions as the array in terms of how ECC is calculated. Only one program is allowed per 128-bit ECC segment.

The UTEST Nonvolatile Memory section is OTP and erase is not allowed. User mode programming of the UTEST section is enabled only when MCR.PEAS is high. The UTEST section is divided into a *Flexible* area where content is managed at System level, and a *Reserved* area where addresses are pre-allocated for storing certain content. The UTEST section may be locked and unlocked against programming via LOCK0.TSLOCK in conjunction with the relative sideband (System specific). For locking purposes, UTEST is treated as a stand-alone block. UTEST access (both Read and Modify) is disabled in Test mode once the Test Mode Disable Seal password is written. Test Flash logical block gets also protected by the Test Mode Disable Seal.

The UTEST section is also set as OPP once the Test Mode Disable Seal password is written, thus programming is only allowed on virgin 128-bit locations.

35.3 Register memory maps and descriptions

In this section some nonvolatile registers are described as well as the usual volatile registers. Note that such entities are not Flip-Flops, but Test Flash block locations with a specific purpose.

During the Flash Initialization phase, the FPEC reads these nonvolatile registers and updates their corresponding volatile registers. When the FPEC detects ECC uncorrectable errors in these special locations, it behaves in the following way:

- In the case of failing system locations (configurations, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a Fatal Error is flagged.
- In the case of failing coherency checks, the volatile registers are filled with ‘1’s and the Flash Initialization ends setting MCR.RE.

35.3.1 Register memory maps

This section presents the memory mapping of all the Flash memory registers.

35.3.1.1 User register memory maps

The Flash User registers represent the communication interface between the host CPU and the FPEC.

Some register bits (command bits) are Read/Write for the CPU and read only for the FPEC. Other register bits (status bits) are Read/Write for the FPEC and read only for the CPU.

The Flash User Registers mapping is shown in *Table 406*.

Table 406. Flash user registers memory map

Address offset	Description	Location
0x0000	Module Configuration Register (MCR)	*Section 35.3.2.1*
0x0004	Alternate Module Configuration Register (MCRA)	*Section 35.3.2.2*
0x0008	Extended Module Configuration Register (MCRE)	*Section 35.3.2.3*
0x000C–0x000F	Reserved	
0x0010	Locking 0 register (LOCK0)	*Section 35.3.2.4*
0x0014	Locking 1 register (LOCK1)	*Section 35.3.2.5*
0x0018	Locking 2 register (LOCK2)	*Section 35.3.2.6*
0x001C	Locking 3 register (LOCK3)	*Section 35.3.2.7*
0x0020–0x0027	Reserved	
0x0028	Alternate Locking 0 register (LOCK0A)	*Section 35.3.2.8*
0x002C	Alternate Locking 1 register (LOCK1A)	*Section 35.3.2.9*
0x0038	Select 0 register (SEL0)	*Section 35.3.2.10*
0x003C	Select 1 register (SEL1)	*Section 35.3.2.11*
0x0040	Select 2 register (SEL2)	*Section 35.3.2.12*
0x0044	Select 3 register (SEL3)	*Section 35.3.2.13*
0x0048–0x004B	Reserved	
0x004C	Program Address Register (PAR)	*Section 35.3.2.14*
0x0050	Address Register (ADR)	*Section 35.3.2.15*
0x0054	User Test register 0 (UT0)	*Section 35.3.2.16*
0x0058	User Multiple Input Signature Register 0 (UM0)	*Section 35.3.2.17.1*
0x005C	User Multiple Input Signature Register 1 (UM1)	*Section 35.3.2.17.2*
0x0060	User Multiple Input Signature Register 2 (UM2)	*Section 35.3.2.17.3*
0x0064	User Multiple Input Signature Register 3 (UM3)	*Section 35.3.2.17.4*
0x0068	User Multiple Input Signature Register 4 (UM4)	*Section 35.3.2.17.5*
0x006C	User Multiple Input Signature Register 5 (UM5)	*Section 35.3.2.17.6*
0x0070	User Multiple Input Signature Register 6 (UM6)	*Section 35.3.2.17.7*
0x0074	User Multiple Input Signature Register 7 (UM7)	*Section 35.3.2.17.8*
0x0078	User Multiple Input Signature Register 8 (UM8)	*Section 35.3.2.17.9*
0x007C	User Multiple Input Signature Register 9 (UM9)	*Section 35.3.2.17.10*
0x0080	Over Program Protection 0 (OPP0)	*Section 35.3.2.18.1*
0x0084	Over Program Protection 1 (OPP1)	*Section 35.3.2.18.2*
0x0088	Over Program Protection 2 (OPP2)	*Section 35.3.2.18.3*

Table 406. Flash user registers memory map (continued)

Address offset	Description	Location
0x008C	Over Program Protection 3 (OPP3)	Section 35.3.2.18.4
0x0090	Test Mode Disable Password Check (TMD)	Section 35.3.2.19
0x0094	HSM ALT-SEL 0 register (ALTSEL0)	Section 35.3.2.20.1
0x0098	HSM ALT-SEL 1 register (ALTSEL1)	Section 35.3.2.20.2
0x0100	Erase Lock Protection 0 (ELOCK0)	Section 35.3.2.21.1
0x0104	Erase Lock Protection 1 (ELOCK1)	Section 35.3.2.21.2
0x0108	Erase Lock Protection 2 (ELOCK2)	Section 35.3.2.21.3
0x010C	Erase Lock Protection 3 (ELOCK3)	Section 35.3.2.21.4
0x0110	Program Lock Protection 0 (PLOCK0)	Section 35.3.2.22.1
0x0114	Program Lock Protection 1 (PLOCK1)	Section 35.3.2.22.2
0x0118	Program Lock Protection 2 (PLOCK2)	Section 35.3.2.22.3
0x011C	Program Lock Protection 3 (PLOCK3)	Section 35.3.2.22.4
0x0134–0x0137	Reserved	

35.3.2 User register descriptions

This section consists of user register descriptions in address order.

35.3.2.1 Module Configuration Register (MCR)

The Module Configuration Register is used to enable and monitor all the Modify operations of the Flash Memory module and to monitor the Read operations.

Address: 0x0000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RRE	AEE	EEE	DWEE				0	0	0	SBC1	SAK	LSW	PEP	PES
W		w1c	w1c	w1c								w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	RE	PEAS	DONE	PEG	PECIE	FERS	0	0	PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 318. Module Configuration Register (MCR)

Table 407. MCR field descriptions

Field	Description
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the analog Read references. If the current and voltage Read references are out of range, this bit is set to signify that previous reads requested may have been corrupted. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A Read Reference Error occurred during a previous Read.
2 AEE	Address Encode Error On every Read request to the Flash, the incoming address is compared to an encoded address (row, column, and block) from the memory array using the read data sense amp timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error is recorded. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An Address Encode Error occurred during a previous Read.
3 EEE	EDC after ECC Error EEE provides information on previous reads monitoring the EDC after ECC feature. On every Read request to the Flash, ECC parity is recomputed (encoding) just after ECC decoding, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error is reported. The EEE flag indicates this event when data is legal or containing up to 3 bit errors. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A previous Read may have been decoded by an ECC logic corrupted.
4:7 DWEE	Double Word Uncorrectable ECC Error DWEE identifies which Double Word is involved in ECC Uncorrectable Error Event. In Code Flash 4 Double Words are read in parallel and ECC is evaluated on a couple of Double Words basis. Therefore DWEE value could be: 1111 if an ECC Uncorrectable Error is found in both couples of Double Words 1100 if an ECC Uncorrectable Error is found in the couple of Double Words at high address 0011 if an ECC Uncorrectable Error is found in the couple of Double Words at low address 0000 if no ECC Uncorrectable Error is found In Data Flash 2 Double Words are read in parallel and ECC is evaluated on a single Double Word basis. Therefore DWEE value could be: 1111 if an ECC Uncorrectable Error is found in both Double Words 1010 if an ECC Uncorrectable Error is found in Double Word at high address 0101 if an ECC Uncorrectable Error is found in Double Word at low address 0000 if no ECC Uncorrectable Error is found DWEE is a read only. Flags bits get cleared when EER is cleared.

Table 407. MCR field descriptions (continued)

Field	Description
11 SBC1	Single Bit Correction 1 SBC1 provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection or ECC Double Error correction, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE1). 0 Reads are occurring normally. 1 An ECC Single Error occurred and was corrected during a previous Read. In Code Flash, with 128b ECC coding, ECC correction is evaluated on couple of Double word basis. In Data Flash for EEE, ECC correction is evaluated on each Double word.
12 SAK	Suspend Acknowledge SAK provides information about the Acknowledge of a suspend request at the setting of DONE: SAK qualifies if the suspend has been accepted (initiated by PSUS or ESUS setting) or if the running command has been completed, since very close to natural end. The aim of this bit is for monitor. The sequence to resume normal functionality (PSUS or ESUS clearing) is the same both is suspend has been acknowledged or not. SAK is read Only 0 Operation has been completed prior that Suspend got processed 1 Suspend Acknowledged
13 LSW	Lock-Sel Write Error LSW provides information about Lock and Sel write denied during the program and erase sequence. If LSW is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Lock and Sel write successful 1 Lock and Sel write not succeeded
14 PEP	Program and Erase Protection Error PEP provides information about program and erase operations with respect to protection errors. A protection error occurs if a program is attempted to a Locked block, or if an erase is selected to a locked block. This is evaluated prior to the operation beginning, and if an error is detected, high voltage operations will not be attempted for this request. If PEP is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program and Erase protection errors do not exist 1 Previous program or erase protection error encountered

Table 407. MCR field descriptions (continued)

Field	Description
15 PES	Program Erase Sequence Error PES provides information about program and erase operations with respect to sequence errors. A sequence error occurs when the program or erase sequence is not followed exactly. If an out-of-order event is detected, PES will assert, and the remainder of the sequence will not be accepted. Sequence errors are checked from the time the PGM or ERS are set (or remains set from previous operation) until EHV is set. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program or Erase phase accepted 1 Program or Erase command encountered an error
16 EER	ECC uncorrectable Event Error EER provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. EER may not be set to ‘1’ by the user. EER is not set for a ECC correctable Error detection and related correction event. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An ECC uncorrectable Error occurred during a previous Read. In Code Flash, with 128b ECC coding, ECC error is evaluated on couple of Double word basis. In Data Flash for EEE, ECC error is evaluated on each Double word.
17 RWE	Read-while-Write event Error RWE provides information on previous Reads during a Modify operation. Read-While-Write Error means that a Read access to the Flash Array partition has occurred while the FPEC was performing a Program or Erase operation or an Array Integrity Check on the same partition. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A RWW Error occurred during a previous Read.
18 SBC	double Bit Correction SBC provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE). 0 Reads are occurring normally. 1 An ECC Double Error occurred and was corrected during a previous Read. In Code Flash, with 128b ECC coding, ECC correction is evaluated on couple of Double word basis. In Data Flash for EEE, ECC correction is evaluated on each Double word.

Table 407. MCR field descriptions (continued)

Field	Description
19 RE	Reset Error This bit provides information on previous resets. Checks are done within the Flash, and if a coherency issue or ECC error is detected during the reset reads used for trimming on the Flash, this status flag will assert. This bit is status only, and is not writable. 0 Flash Boot phase has occurred normally. 1 A reset error has been encountered.
20 PEAS	Program/Erase Access Space PEAS is used to indicate which space is valid for Program and Erase operations: main array space or UTEST section within Test Flash block. – PEAS is captured and held with the first Interlock Write done for Modify operations. – PEAS is retained between sampling events (that is subsequent to first interlock writes). – PEAS is changed during erase-suspended program (when program is in UTEST) and reverts back to its original state (related to erase suspended operation) once the erase-suspended program is completed (at MCR.PGM 1-0 transition). 0 UTEST address space is disabled for program and main address space enabled. 1 UTEST address space is enabled for program and main address space disabled.
21 DONE	Modify operation DONE DONE indicates if the Flash Memory module is performing a high voltage operation. DONE is set to ‘1’: – On termination of the Flash Memory module reset. – At the end of program and Erase High Voltage sequences. – After “1-0” transition of EHV, which aborts a high voltage Erase/Program operation (within a time equal or below to t_{ESUS}/t_{PSUS} that are Erase/Program Suspend Latency). – After “0-1” transition of ESUS/PSUS which suspends an Erase/Program operation (within t_{ESUS}/t_{PSUS} time equal to the Erase/Program Suspend Latency). DONE is cleared to ‘0’: – just after “0-1” transition of EHV, which initiates a high voltage operation, or after resuming a suspended operation. 0 Flash is executing a high voltage operation. 1 Flash is not executing a high voltage operation.

Table 407. MCR field descriptions (continued)

Field	Description
22 PEG	Program/Erase Good PEG is automatically updated to show the completion status of the last Flash Program or Erase sequence involving high voltage operations. PEG is evaluated on the completion status of only the interlocked doubleword during page program or quad-page programming where some doubleword addresses are not interlocked (already programmed or intended to remain virgin). PEG is set to ‘0’ after Aborting a Program/Erase high voltage operation, indicating sequence failure. PEG is set to ‘1’ when the Flash Memory module is reset, unless a Flash Initialization error has been detected. **Note:** PEG is valid only when PGM=1 or ERS=1, or both, and after DONE 0-1 transitions due to an Abort or the completion of a Program/Erase operation. PEG remains valid until PGM/ERS 1-0 transitions or EHV 0-1 transitions. PEG is not valid after DONE 0-1 transitions when ESUS/PSUS is set to logic ‘1’. If Program or Erase is attempted on blocks that are locked, PEG=1 indicating that the content of the block were properly protected from the Program or Erase operation. Blocks can be locked through LOCK0/1/2/3 registers OR with flh_plock0/1/2/3 sideband signals. If a Program operation tries to write ‘1’ to bits that are ‘0’, the program operation is correctly executed on the new bits to be programmed at ‘0’, but PEG is cleared indicating that the requested operation has failed. If Programming during an Erase-Suspend a location that was under suspended erase, program is not executed and PEG is set to ‘0’. With end2end ECC scheme, during an Interlock Write also ECC parity bits are input. After setting EHV, the FPEC checks the integrity of the data received in terms of ECC. If any doubleword in the doubleword program, in the page program or in the quad-page program, contains an inconsistency between data and parity, the whole operation is rejected and a PEG=0 is set. PEG behavior with blocks marked as OPP (though flh_otpen0/1/2/3 sideband signals) is the following: PEG=0 if the location to be programmed is not virgin, and thus would not be over programmed. In Array Integrity Check or Margin Read PEG is not altered when the operation is completed, regardless of the occurrence of any error, abort, break point or suspend. The presence of errors can only be detected by comparing the checksum value stored in UM0-9 and by checking the MCR flags. 0 Program, Erase operation failed or Program, Erase aborted. 1 Program or Erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag. 0 An interrupt request is not generated when the DONE flag is set. 1 An interrupt request is generated when the DONE flag is set.

Table 407. MCR field descriptions (continued)

Field	Description
24 FERS	Factory Mode FERS is used for Erase and Program operations to enable a faster operation. Factory Mode must adhere to *Initial maximum (All Temperatures)* conditions, with respect to voltage conditions and Write/Erase cycling. The temperature can range from -40 °C to 150 °C. Factory Mode does not support Read and RWW operations. – FERS may be set to enable a Factory Erase or Program after setting ERS or PGM but prior to setting EHV. – FERS can be cleared only when EHV is low and DONE is high. – FERS is cleared on reset **Note:** Factory Mode may be permanently disabled by writing a location in the UTEST Nonvolatile Memory space. Offset 0x0020 in the UTEST Nonvolatile Memory space is checked at reset, and if programmed with data that contains at least one zero, a Factory Mode is not allowed, and FERS is locked. Writes attempted that violate the above cases result in FERS remaining cleared. 0 Factory Mode is disabled. 1 Factory Mode is enabled.
27 PGM	Program PGM is used to setup the Flash Memory module for a Program operation. – “0-1” transition of PGM initiates a program sequence. – “1-0” transition of PGM ends the program sequence. PGM can only be set in the following conditions: – User Mode Read (ERS is low and UT0.UTE is low); – Erase Suspend (ERS and ESUS high) with EHV low. PGM can only be cleared by the user when PSUS and EHV are low and DONE is high. PGM is cleared on reset. **Note:** In an erase-suspended program, programming Flash memory locations in blocks which were being operated on in the Erase is forbidden. Command is not accepted and PEG is set to ‘0’ 0 Flash is not executing a program sequence. 1 Flash is executing a program sequence.

Table 407. MCR field descriptions (continued)

Field	Description
28 PSUS	Program Suspend PSUS is used to indicate that the Flash Memory module is in Program Suspend or in the process of entering a Suspend state. The Flash Memory module is in Program Suspend when PSUS=1 and DONE=1. – PSUS can be set high only when PGM and EHV are high. – "0-1" transition of PSUS starts the sequence which sets DONE and places the Flash in Program Suspend. The Flash Memory module enters Suspend within t_{PSUS} of this transition. – PSUS can be cleared only when DONE and EHV are high. – "1-0" transition of PSUS with EHV=1 starts the sequence which clears DONE and returns the module to Program. The Flash Memory module cannot exit Program Suspend and clear DONE while EHV is low. In the case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), PSUS has to be cleared first in order to resume currently higher level suspended operation that is program. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.
29 ERS	Erase ERS is used to setup the Flash Memory module for an Erase operation. – "0-1" transition of ERS initiates an Erase sequence. – "1-0" transition of ERS ends the Erase sequence. ERS can be set only under User Mode Read (PGM is low and UT0.UTE is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash is not executing an Erase sequence. 1 Flash is executing an Erase sequence.

Table 407. MCR field descriptions (continued)

Field	Description
30 ESUS	Erase Suspend ESUS is used to indicate that the Flash Memory module is in Erase Suspend or in the process of entering a Suspend state. The Flash Memory module is in Erase Suspend when ESUS = 1 and DONE=1. – ESUS can be set high only when ERS and EHV are high and PGM is low. – “0-1” transition of ESUS starts the sequence which sets DONE and places the Flash in Erase Suspend. The Flash Memory module enters Suspend within t_{ESUS} of this transition. – ESUS can be cleared only when DONE and EHV are high and PGM is low. – “1-0” transition of ESUS with EHV=1 starts the sequence which clears DONE and returns the module to Erase. The Flash Memory module cannot exit Erase Suspend and clear DONE while EHV is low. In case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), ESUS cannot be cleared first. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.

Table 407. MCR field descriptions (continued)

Field	Description
31 EHV	Enable High Voltage The EHV bit enables the Flash Memory module for a high voltage Program/Erase operation. EHV is cleared on reset. EHV must be set after an Interlock Write to start a Program/Erase sequence. EHV may be set under one of the following conditions: – Erase (ERS=1, ESUS=0, UT0.AIE=0) – Program (ERS=0, ESUS=0, PGM=1, UT0.AIE=0) – Erase-suspended program (ERS=1, ESUS=1, PGM=1, PSUS=0, UT0.AIE=0). For a program operation to initiate while an Erase is suspended, EHV must be cleared while in Erase-Suspend (after DONE transitions to 1) before setting PGM. In normal operation, a “1-0” transition of EHV with DONE high, PSUS low and ESUS low terminates the current Program/Erase high voltage operation. In an erase-suspended program operation, a “1-0” transition of EHV with DONE, ESUS, PGM and ERS high and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a “1-0” transition of EHV with DONE low and the eventual Suspend bit for the current program/erase operation low. An Abort causes the value of PEG to be cleared, indicating failed Program/Erase. Address locations being operated on by the aborted operation contain indeterminate data after an Abort. EHV cannot be set to ‘1’ after an Abort request (EHV being cleared) until the Abort sequence is completed (DONE transitions to 1). Once the Abort sequence is completed (DONE=1) a new operation of the kind aborted can start provided that a new interlock is given and then EHV is set to ‘1’ again. A suspended operation cannot be aborted. Aborting a high voltage operation leaves the Flash Memory module addresses in an indeterminate data state. This may be recovered by executing an Erase on the affected blocks. EHV may be written during Suspend. EHV must be high to exit Suspend, resuming previously suspended command. EHV cannot not be written during transition into Suspend state (after Suspend bit (ESUS or PSUS) is set high) until DONE transitions to ‘1’. EHV may not be written after ESUS is set and before DONE transitions high. EHV may not be cleared after ESUS is cleared and before DONE transitions low. 0 Flash is not enabled to perform a high voltage operation. 1 Flash is enabled to perform a high voltage operation.

A number of MCR bits are protected against Write when another bit, or set of bits, are in a specific state. These Write locks are covered on a bit-by-bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

The Flash Memory module does not allow the user to write bits simultaneously (which puts the device into an illegal state). This is implemented through a priority mechanism among the bits shown in *Table 408*.

Table 408. MCR Bits Set/Clear Priority Levels

Priority level	MCR bits
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If the user attempts to write two or more MCR bits simultaneously then only the bit with the lowest priority level is written.

35.3.2.2 Alternate Module Configuration Register (MCRA)

The Alternate Module Configuration Register is used to enable and monitor all the Modify operations of the Flash Memory module managed through Alternate interface.

Address: 0x0004 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	SAKA	LSWA	PEPA	PESA
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PEASA	DONEA	PEGA	0	0	0	0	PGMA	PSUSA	ERSA	ESUSA	EHVA
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 319. Alternate Module Configuration Register (MCRA)

Table 409. MCRA field descriptions

Field	Description
12 SAKA	Alternate Suspend Acknowledge SAKA provides information about the Acknowledge of a suspend request at the setting of DONEa: SAKA qualifies if the suspend has been accepted (initiated by PSUSa or ESUSa setting) or if the running command has been completed, since very close to natural end. The aim of this bit is for monitor. The sequence to resume normal functionality (PSUSa or ESUSa clearing) is the same both is suspend has been acknowledged or not. SAKA is read Only 0 Operation has been completed prior that Suspend got processed. 1 Suspend Acknowledged
13 LSWA	Alternate Lock Write Error LSWA provides information about Lock write denied during the program and erase sequence. If LSWA is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Lock and Sel write successful. 1 Lock and Sel write not succeeded.
14 PEPA	Alternate Program and Erase Protection Error PEPA provides information about program and erase operations with respect to protection errors. A protection error occurs if a program is attempted to a Locked block, or if an erase is selected to a locked block. This is evaluated prior to the operation beginning, and if an error is detected, high voltage operations will not be attempted for this request. If PEPA is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program and Erase protection errors do not exist. 1 Previous program or erase protection error encountered.
15 PESA	Alternate Program Erase Sequence Error PESA provides information about program and erase operations with respect to sequence errors. A sequence error occurs when the program or erase sequence is not followed exactly. If an out-of-order event is detected, PESA will assert, and the remainder of the sequence will not be accepted. Sequence errors are checked from the time the PGMA or ERSA are set (or remains set from previous operation) until EHVA is set. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program or Erase phase accepted 1 Program or Erase command encountered an error
20 PEASA	Program/Erase Access Space PEASA is used to indicate which space is valid for Program and Erase operations: main array space or UTEST block. It should be noted that erase is not allowed to the UTest NVM space, therefore If an interlock write is done to the UTest NVM space with ERSA set, PEASA will remain 0. See PEAS definition in the MCR register for more information(*Table 407*).
21 DONEA	Modify operation DONE DONEA indicates if the embedded Flash memory is performing a high-voltage operation for the Alternate program and erase interface. See DONE definition in the MCR register for more information (*Table 407*).

Table 409. MCRA field descriptions (continued)

Field	Description
22 PEGA	Program/Erase Good The PEGA bit indicates the completion status of the last Flash memory program or erase sequence for which high-voltage operations were initiated for the Alternate program and erase interface. See PEG definition in the MCR register for more information (*Table 407*). It should be noted that in the event of a error on the alternate interface PEGA will clear, but the ADR register will not be updated with the failing location. The ADR register is only valid for main interface program or erase fails, and other fails noted in the ADR register.
27 PGMA	Program PGMA is used to set up embedded Flash memory for a program operation for the Alternate program and erase interface. See PGM definition in the MCR register for more information (*Table 407*).
28 PSUSA	Program Suspend PSUSA is used to indicate the embedded Flash memory is in program suspend or in the process of entering a suspend state for the Alternate program and erase interface. See PSUS definition in the MCR register for more information (*Table 407*).
29 ERSA	Erase ERSA is used to set up embedded Flash memory for an erase operation for the Alternate program and erase interface. See ERS definition in the MCR register for more information (*Table 407*).
30 ESUSA	Erase Suspend ESUSA is used to indicate that the embedded Flash memory is in erase suspend or in the process of entering a suspend state for the Alternate program and erase interface. See ESUS definition in the MCR register for more information (*Table 407*).
31 EHVA	Enable High Voltage The EHVA bit enables the embedded Flash memory for a high-voltage program/erase operation for the Alternate program and erase interface. See EHV definition in the MCR register for more information (*Table 407*).

A number of MCR bits are protected against Write when another bit, or set of bits, are in a specific state. These Write locks are covered on a bit-by-bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

35.3.2.3 Extended Module Configuration Register (MCRE)

The Extended Module Configuration Register value is based on the size, blocks and address space configuration of the Flash Memory module.

Address: 0x0008 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	N128KL		N256K					N64KH			N32KH		N16KH		
W																
Reset	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	N64KM			N32KM		N16KM			N64KL			N32KL		N16KL		
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	1

Figure 320. Extended Module Configuration Register (MCRE)

Table 410. MCRE field descriptions

Field	Description
1:2 N128KL	Number of 128 KB blocks in low space. 00 Zero 128 KB block. 01 One 128 KB block. 10 Two 128 KB blocks. 11 Three 128 KB blocks.
3:7 N256K	Number of 128 KB and 256 KB blocks in the 256K space The value of the N256K field is dependent upon the size of the embedded Flash memory. N256K is read only. N256K value represents in the number of blocks effectively available. 00000 Zero block. 00001 One block. 00010 Two blocks. 00011 Three blocks. 00100 Four blocks. 00101 Five blocks. 00110 Six blocks. 00111 Seven blocks 01000 Eight blocks. 01001 Nine blocks. 01010 Ten blocks. … 11111 thirty-one blocks.
8:10 N64KH	Number of 64 KB blocks in high space N64KH is read only. 000 Zero 64 KB block. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.

Table 410. MCRE field descriptions (continued)

Field	Description
11:12 N32KH	Number of 32 KB blocks in high space N32K is read only. 00 Zero 32 KB block. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13:15 N16KH	Number of 16 KB blocks in high space N16K is read only. 000 Zero 16 KB block. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16:18 N64KM	Number of 64 KB blocks in mid space N64KM is read only. 000 Zero 64 KB block. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19:20 N32KM	Number of 32 KB blocks in mid space N32KM is read only. 00 Zero 32 KB block. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21:23 N16KM	Number of 16 KB blocks in mid space N16KM is read only. 000 Zero 16 KB block. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.

Table 410. MCRE field descriptions (continued)

Field	Description
24:26 N64KL	Number of 64 KB blocks in low space N64KL is read only. 000 Zero 64 KB block. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Three 64 KB blocks + One 96KB block. 110 Reserved. 111 Reserved.
27:28 N32KL	Number of 32 KB blocks in low space N32KL is read only. 00 Zero 32 KB block. 01 Two 32 KB blocks. 10 One 32 KB block. 11 Three 32 KB blocks.
29:31 N16KL	Number of 16 KB blocks in low space N16KL is read only. 000 Zero 16 KB block. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Three 16 KB blocks. 110 Five 16 KB blocks. 111 Reserved.

35.3.2.4 Lock 0 register (LOCK0)

The LOCK0 (Low/Mid Address Space Block Locking) register provides a means of protecting blocks from being modified. LOWLOCK and MIDLOCK bits corresponding to non existent blocks are read as ‘1’.

To determine the effective status of the Low/Mid Address Space, combine (Or) LOCK0 with sidebands:

- flh_plock0 – in order to evaluate protection against programming.
- flh_elock0 – in order to evaluate protection against erase.

The effect of programming or erasing a block protected with LOCK0 is that no modification occurs and PEG is set to ‘1’.

Address: 0x0010 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCK	0	LOWLOCK													
W																
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCK															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 321. Lock 0 register (LOCK0)

Table 411. LOCK0 field descriptions

Field	Description
0 TSLOCK	UTEST NVM block lock This bit is used to lock the UTEST NVM from programming (UTEST NVM is not erasable). The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the TSLOCK bit is locked. The effective status of the UTEST protection is done combining (ORed) LOCK0.TSLOCK with sidebands *flh_plock0.31* in order to evaluate protection against program. 0 Test Address Space Block is unlocked and can be modified. 1 Test Address Space Block is locked and cannot be modified.

Table 411. LOCK0 field descriptions (continued)

Field	Description
2:15 LOWLOCK	Low address space block lock These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the LOWLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the LOWLOCK bits default to locked and are not writable. The reset value is always '1' and register writes have no effect. 0 Low Address Space Block is unlocked and can be modified. 1 Low Address Space Block is locked and cannot be modified.
16:31 MIDLOCK	Mid address space block lock These bits are used to lock the blocks of Mid Address Space from Program and Erase. The block numbering for the Mid Blocks starts at MIDLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the MIDLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the MIDLOCK bits default to locked, and are not writable. The reset value is always '1' and register writes have no effect. 0 Mid Address Space Block is unlocked and can be modified. 1 Mid Address Space Block is locked and cannot be modified.

35.3.2.5 Lock 1 register (LOCK1)

The LOCK1 (High/Data Address Space Block Locking) register provides a means of protecting blocks from being modified. HIGHLOCK bits corresponding to nonexistent blocks on are read as ‘1’.

- To determine the effective status of the High Address Space, combine (Or) LOCK1 with sidebandsflh_plock1 – in order to evaluate protection against program
- flh_elock1 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK1 is that no modification occurs and PEG is set to ‘1’.

Address: 0x0014 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HIGHLOCK															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 322. Lock 1 register (LOCK1)

Table 412. LOCK1 field descriptions

Field	Description
16:31 HIGHLOCK	High address space block lock These bits are used to lock the blocks of High Address Space from Program and Erase. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the HIGHLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). When blocks are not present (due to configuration or total memory size), the HIGHLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 High Address Space Block is unlocked and can be modified. 1 High Address Space Block is locked and cannot be modified.

35.3.2.6 Lock 2 Register (LOCK2)

The LOCK2 (256K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A256KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the A256KLOCK Address Space, combine (Or) LOCK2 with sidebands:

- flh_plock2 – in order to evaluate protection against program
- flh_elock2 – in order to evaluate protection against erase

The effect of programming/erasing a block protected with LOCK2 is that no modification occurs and PEG is set to ‘1’.

Address: 0x0018 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	A256KLOCK															
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KLOCK															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 323. Lock 2 Register (LOCK2)

Table 413. LOCK2 field descriptions

Field	Description
0:31 A256KLOCK	256K address space block lock A256KLOCK[31:0] These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the 256 KB blocks starts with A256KLOCK[0] and continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the A256KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A256KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 A256K Address Space Block is unlocked and can be modified. 1 A256K Address Space Block is locked and cannot be modified.

35.3.2.7 Lock 3 Register (LOCK3)

The LOCK3 (256K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A256KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the A256KLOCK Address Space, combine (Or) LOCK3 with sidebands:

- flh_plock3 – in order to evaluate protection against program
- flh_elock3 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK3 is that no modification occurs and PEG is set to ‘1’.

Address: 0x001C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	A256KLOCK															
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KLOCK															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 324. Lock 3 Register (LOCK3)

Table 414. LOCK3 field descriptions

Field	Description
0:31 A256KLOCK	256K address space block lock A256KLOCK[63:32] These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the 256 KB blocks starts with A256KLOCK[32] and continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the A256KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A256KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 A256K Address Space Block is unlocked and can be modified. 1 A256K Address Space Block is locked and cannot be modified.

35.3.2.8 Alternate Lock 0 register (LOCK0A)

The Alternate LOCK0A (Low/Mid Address Space Block Locking) register provides a means of protecting blocks from being modified on the alternate program and erase interface. LOWLOCKA and MIDLOCKA bits corresponding to non existent blocks are read as ‘1’.

The effective status of the Alternate Low Address Space is not affected by sidebands as in the Main Interface

The effect of programming/erasing a block protected with LOCK1A is that no modification occurs and PEGA is set to ‘1’.

Address: 0x0028 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCKA	0	LOWLOCKA													
W																
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCKA															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 325. Alternate Lock 0 register (LOCK0A)

Table 415. LOCK0A field descriptions

Field	Description
0 TSLOCKA	Alternate UTEST NVM block lock This bit is used to lock the UTEST NVM from programming (UTEST NVM is not erasable). The LOCK field is not writable: – Once an Interlock Write is completed until MCRA.DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the TSLOCK bit is locked. The effective status of the UTEST protection is done combining (ORed) LOCK0A.TSLOCKA with sidebands *flh_aplock_utest* in order to evaluate protection against program. 0 The UTEST Block is unlocked and can be modified on the alternate interface. 1 The UTEST Block is locked and cannot be modified on the alternate interface.

Table 415. LOCK0A field descriptions (continued)

Field	Description
2:15 LOWLOCKA	Alternate Low address space block lock These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the Low Blocks starts at LOWLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the LOWLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the LOWLOCKA bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Alternate Low Address Space Block is unlocked and can be modified. 1 Alternate Low Address Space Block is locked and cannot be modified.
16:31 MIDLOCKA	Alternate Mid address space block lock These bits are used to lock the blocks of Mid Address Space from Program and Erase. The block numbering for the Mid Blocks starts at MIDLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the MIDLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the MIDLOCKA bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Alternate Mid Address Space Block is unlocked and can be modified. 1 Alternate Mid Address Space Block is locked and cannot be modified.

35.3.2.9 Alternate Lock 1 register (LOCK1A)

The Alternate LOCK1A (Alternate High Address Space Block Locking) register provides a means of protecting blocks from being modified on the alternate program and erase interface. HIGHLOCK bits corresponding to nonexistent blocks on are read as ‘1’.

The effect of programming/erasing a block protected with LOCK1a is that no modification occurs and PEGa is set to ‘1’.

Address: 0x002C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HIGHLOCKA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 326. Alternate Lock 1 register (LOCK1A)

Table 416. Alternate LOCK1A field descriptions

Field	Description
16:31 HIGHLOCKA	Alternate High address space block lock These bits are used to lock the blocks of High Address Space from Program and Erase. The block numbering for the High Blocks starts at HIGHLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the HIGHLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). When blocks are not present (due to configuration or total memory size), the HIGHLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 High Address Space Block is unlocked and can be modified. 1 High Address Space Block is locked and cannot be modified.

35.3.2.10 Select 0 register (SEL0)

The SEL0 (Low/Mid Address Space Block Select 0) register provides a means to select blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL0 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 35.4.5.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL0 at the start of the Erase operation) while writing has no effect.

Address: 0x0038 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWSEL													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 327. Select 0 register (SEL0)

Table 417. SEL0 field descriptions

Field	Description
2:15 LOWSEL	Low address space block Select The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding LOWSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 Low Address Space Block is unselected for Erase. 1 Low Address Space Block is selected for Erase.

Table 417. SEL0 field descriptions (continued)

Field	Description
16:31 MIDSEL	Mid address space block Select The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size), the corresponding MIDSEL bits default to unselected and are not writable.The reset value is always ‘0’, and register writes have no effect. 0 Mid Address Space Block is unselected for Erase. 1 Mid Address Space Block is selected for Erase.

35.3.2.11 Select 1 register (SEL1)

The SEL1 (High Address Space Block Select Register 1) provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL1 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 35.4.5.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL1 at the start of the Erase operation) while writing has no effect.

Address: 0x003C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HIGHSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 328. Select 1 register (SEL1)

Table 418. SEL1 field descriptions

Field	Description
16:31 HIGHSEL	High/Data address space block Select 1 The block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding HIGHSEL bits default to unselected and are not writable. The reset value is always '0' and register writes have no effect. 0 High Address Space Block is unselected for Erase. 1 High Address Space Block is selected for Erase.

35.3.2.12 Select 2 register (SEL2)

The SEL2 (256K Address Space Block Select 2) register provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL2 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 35.4.5.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL2 at the start of the Erase operation) while writing has no effect.

Address: 0x0040 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 329. Select 2 register (SEL2)

Table 419. SEL2 field descriptions

Field	Description
0:31 A256KSEL	256K address space block select A256KSEL[31:0] The block numbering for the 256 KB blocks starts with A256KSEL[0] and continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding A256KSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 A256K Address Space Block is unselected for Erase. 1 A256K Address Space Block is selected for Erase.

35.3.2.13 Select 3 register (SEL3)

The SEL3 (256K Address Space Block Select 3) register provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL3 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 35.4.5.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL3 at the start of the Erase operation) while writing has no effect.

Address: 0x0044 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 330. Select 3 register (SEL3)

Table 420. SEL3 field descriptions

Field	Description
0:31 A256KSEL	256K address space block select A256KSEL[63:32] The block numbering for the 256 KB blocks starts with A256KSEL[32] and continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding A256KSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 A256K Address Space Block is unselected for Erase. 1 A256K Address Space Block is selected for Erase.

35.3.2.14 Program Address register (PAR)

The Program Address register provides the last successful address programmed. A successful program is defined as a program operation of data that is not all 1s, to an unlocked block and the operation results in MCR.PEG returning a ‘1’. This value is held until the next successful address program event.

Address: 0x004C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PAR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PAR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 331. Program Address register (PAR)

Table 421. PAR field descriptions

Field	Description
8:28 PAR	Address The Address Register provides the last successful address programmed through the Main Interface. Address given is in Flash format and has to be intended as in the Flash memory Map: offset respect to absolute address. Mapping can be found in *Section 35.3.2.15: Address register (ADR)*

35.3.2.15 Address register (ADR)

The Address register provides the first failing address in the event of module failures (ECC, RWW or FPEC). ADR captures the first failing address for the modes listed in *Table 423* according to the priority rules in *Table 424*.

In case of Array Integrity Check or Margin Read with Break Point enabled (UT0.AIBPE=1), the ADR value matches the address where the Break Point was triggered only when the current run (first start or re-start after a breakpoint (NAIBP cleared)) starts with MCR.ERR and MCR.SBC flags cleared, otherwise the ADR priority rule overrides Break Point.

Address: 0x0050 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAD	0	0	0	0	0	0	0	ADDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 332. Address register (ADR)

Table 422. ADR field descriptions

Field	Description
0 SAD	Test Address When this bit is high, the address indicated by ADDR[23:3] belongs to the Test Block.
8:28 ADDR	Address The Address Register provides the first failing address in the event of ECC Uncorrectable error (MCR.EER set) or the first failing address in the event of RWW error (MCR.RWE set), or the address of a failure that may have occurred in a FPEC operation (MCR.PEG cleared). The Address Register also provides the first address at which a ECC correction occurs (MCR.SBC or MCR.SBC1, or both are set) if the SoC is configured to show this feature through UT0.SBCE or UT0.SBCE1, or both. The ECC uncorrectable error detection takes the highest priority, followed by the RWW error, the FPEC error and the ECC error corrections. When accessed, ADR provides the address related to the first event occurred with the highest priority. In case of a simultaneous ECC Events Detection, bits ADDR[4] and ADDR[3] output ‘0’. In User mode the Address Register is read only. Address given is in Flash format and has to be intended as in the Flash memory Map.

Table 423. ADR update mode list

Mode	Update on error	Note
Read	ECC Uncorrectable Error Detection, ECC Double Error Correction (if UT0.SBCE=1) ECC Single Error Correction (if UT0.SBCE1=1)	—
Modify	FPEC error	—
RWW	Read-While-Write error	In RWW all Read and Modify cause of update on error may also occur
User Test	ECC Uncorrectable Error Detection, ECC Double Error Correction (if UT0.SBCE=1) ECC Single Error Correction (if UT0.SBCE1=1)	With Break Point enabled (UT0.AIBPE=1), ADR requires MCR.ERR and MCR.SBC to be cleared

Table 424. ADR content: priority list

Priority level	Error flag	ADR content
1	MCR.EER = 1	Address of first ECC Uncorrectable Error
2	MCR.RWE = 1	Address of first RWW Error
3	MCR.PEG = 0	Address of first FPEC Error
4	MCR.SBC = 1	Address of first ECC Double Error Correction
5	MCR.SBC1 = 1	Address of first ECC Single Error Correction

Table 425. Flash memory partition 0 addresses mapping

Block	Flash format addresses	Absolute addresses
B0F0	0x0000_0000 to 0x0000_3FFF	0x00FC_0000 to 0x00FC_3FFF
B0F1	0x0000_4000 to 0x0000_7FFF	0x00FC_4000 to 0x00FC_7FFF
B0F2	0x0000_8000 to 0x0000_FFFF	0x00FC_8000 to 0x00FC_FFFF
B0F3	0x0001_0000 to 0x0001_7FFF	0x00FD_0000 to 0x00FD_7FFF
B0F4	0x0021_8000 to 0x0021_FFFF	0x00FD_8000 to 0x00FD_FFFF
B0F5	0x0002_0000 to 0x0003_FFFF	0x00FE_0000 to 0x00FF_FFFF
B0F6	0x0088_0000 to 0x008B_FFFF	0x0100_0000 to 0x0103_FFFF
B0F7	0x0004_0000 to 0x0007_FFFF	0x0104_0000 to 0x0107_FFFF
B0F8	0x0008_0000 to 0x000B_FFFF	0x0108_0000 to 0x010B_FFFF
B0F9	0x000C_0000 to 0x000F_FFFF	0x010C_0000 to 0x010F_FFFF
B0F10	0x0010_0000 to 0x0013_FFFF	0x0110_0000 to 0x0113_FFFF
B0F11	0x0014_0000 to 0x0017_FFFF	0x0114_0000 to 0x0117_FFFF
B0F12	0x0018_0000 to 0x001B_FFFF	0x0118_0000 to 0x011B_FFFF
B0F13	0x0080_C000 to 0x0080_FFFF	0x0060_C000 to 0x0060_FFFF
B0F14	0x0022_0000 to 0x0023_FFFF	0x0040_0000 to 0x0040_3FFF

Table 425. Flash memory partition 0 addresses mapping (continued)

Block	Flash format addresses	Absolute addresses
B0UT	0x0080_0000 to 0x0080_3FFF	0x0040_0000 to 0x0040_3FFF
B0BF	0x0080_4000 to 0x0080_7FFF	0x0040_4000 to 0x0040_7FFF

Table 426. Flash Memory EEPROM data partition 1 addresses mapping

Block	Flash format addresses	Absolute addresses
D0_00	0x0090_0000 to 0x0090_3FFF	0x0080_0000 to 0x0080_3FFF
D0_01	0x0090_4000 to 0x0090_7FFF	0x0080_4000 to 0x0080_7FFF
D0_02	0x0090_8000 to 0x0090_BFFF	0x0080_8000 to 0x0080_BFFF
D0_03	0x0090_C000 to 0x0090_FFFF	0x0080_C000 to 0x0080_FFFF

Table 427. Flash Memory HSM EEPROM Data Partition 1 addresses mapping

Block	Flash format addresses	Absolute addresses
H0_00	0x0098_0000 to 0x0098_3FFF	0x0068_0000 to 0x0068_3FFF
H0_01	0x0098_4000 to 0x0098_7FFF	0x0068_4000 to 0x0068_7FFF

35.3.2.16 User Test 0 register (UT0)

The User Test 0 register feature gives the user of the Flash Memory module the ability to perform test features on the Flash.

The User Test 0 register allows control of the way the Flash content check is performed. Bits MRE, MRV and AIS of the User Test 0 register are not accessible when MCR.DONE or UT0.AID are low. Reading returns indeterminate data while writing has no effect.

Address: 0x0054 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UTE	SBCE	0	0	0	0	0	0	SBCE1	0	0	0	0	CPR	CPA	CPE
W	UTE	SBCE							SBCE1					CPR	CPA	CPE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	VTD	0	0	NAIBP	AIBPE	0	AISUS	MRE	MRV	0	AIS	AIE	AID
W				VTD			NAIBP	AIBPE		AISUS	MRE	MRV		AIS	AIE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 333. User Test 0 register (UT0)

Table 428. UT0 field descriptions

Field	Description
0 UTE	User Test Enable This status bit gives indication when User Test is enabled. All bits in UT0 and UM0–UM9 are locked when this bit is ‘0’. This bit is not writable to a ‘1’, but may be cleared. The reset value is ‘0’. To set UTE, write the password 0xF9F99999 to the UT0 register. The UTE password is only accepted if PGM=0 and ERS=0 (Program and Erase are not being requested). UTE can only be cleared if AID=1, MRE=0 and AIE=0. While clearing UTE, writes to set AIE or MRE are ignored.
1 SBCE	Double Bit Correction Enable This bit, when high, enables to flag the information about any eventual ECC Double Bit Correction in the Flash array (MCR.SBC and ADR). This is true in read mode and in Array Integrity Check. SBCE is also used as an Enable for interrupt signals created by the embedded Flash memory. ECC corrections that occur when SBCE is cleared are not logged.
8 SBCE1	Single Bit Correction Enable This bit, when high, enables to flag the information about any eventual ECC Single Bit Correction in the Flash array (MCR.SBC1 and ADR). This is true in read mode and in Array Integrity Check. SBCE1 is also used as an Enable for interrupt signals created by the embedded Flash memory. ECC corrections that occur when SBCE1 is cleared are not logged.
13 CPR	Customer Programmable Read Voltage and Reference Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block. If this bit is set and the Customer Programmable Detection location in UTEST is read, a Read Voltage and Reference Error detection is set. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled.
14 CPA	Customer Programmable Address Encode Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block. If this bit is set, and the Customer Programmable Detection location in UTEST is read an Address Encode Error detection is set. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block. If this bit is set, and the Customer Programmable Detection location in UTEST is read a EDC after ECC Error detection is set. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled

Table 428. UT0 field descriptions (continued)

Field	Description
19 VTD	Vth Distribution To enable Vth Distribution, VTD may be set. Effective setting of this bit is allowed when SoC has been brought into Failure Analysis Mode. When Vth Distribution is enabled, multiple run are needed to assess the Vth status of the cells belonging to selected blocks. At the end of each run NAIPB has to be checked and if equals to '1' it has to be cleared to start an increased voltage step run. See NAIBP description for more information. 0 Vth Distribution is not enabled. 1 Vth Distribution is enabled.
22 NAIBP	Next Array Integrity Break Point If AIBPE is set, NAIBP is set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to '1' but may be cleared to '0'. Clearing NAIBP to '0' enables the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID is set with NAIBP remaining '0'. If VTD is set, NAIBP is set once a voltage step run has been completed during the voltage sweep in Vth Distribution. NAIBP is not writable to '1' but may be cleared to '0'. Clearing NAIBP to '0' enables the Vth Distribution run yo be re-started with an increased voltage step. If the Vth Distribution completes the last voltage step run, AID is set with NAIBP remaining '0'. 0 Array Integrity-Vth Distribution is not currently at a break point. 1 Array Integrity-Vth Distribution is at a break point.
23 AIBPE	Array Integrity Break Point Enable To enable breakpoints during an Array Integrity test, AIBPE may be set. When breakpoints are enabled, if the run has been started with MCR.ERR (and MCR.SBC when UT0.SBCE=1) flag cleared, on the eventual occurrence of a breakpoint condition, MCR flags and ADR value may be updated. See NAIBP description for more information about Array Integrity breakpoints. 0 Array Integrity breakpoints are not enabled. 1 Array Integrity breakpoints are enabled during Array Integrity Checks.
25 AISUS	Array Integrity Suspend AISUS toggles Suspend of an Array Integrity - Margin Read Operation. AISUS can be written to '1' only when AID is low AISUS can be written to '0' only when AID is high. Attempting to Write AISUS and AIE on the same clock cycle results in only AIE being written. 0 Array Integrity sequence not suspended. 1 Array Integrity sequence is suspended.
26 MRE	Margin Read Enable MRE enables margin reads to be done. MRE combined with MRV enables User Margin mode reads to be done. Margin reads run on a block-by-block basis in linear address sequence selected by SEL0, SEL1 and SEL2. This bit is not accessible when MCR.DONE or UT0.AID are low or if AISUS or NAIBP are high: reading returns indeterminate data while writing has no effect. 0 Margin Read is not enabled. 1 Margin Read is enabled.

Table 428. UT0 field descriptions (continued)

Field	Description
27 MRV	Margin Read Value If MRE is high, MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). This bit is not accessible whenever MCR.DONE or UT0.AID are low or if AISUS or NAIBP are high: reading returns indeterminate data while writing has no effect. 0 Zero's (programmed) margin reads are requested (if MRE=1). 1 One's (erased) margin reads are requested (if MRE=1).
29 AIS	Array Integrity Sequence AIS determines the address sequence to be used during Array Integrity checks. The default AIS=0 checks the read propagation paths along proprietary sequences followed by user code. The sequential run sequence time is significantly shorter than the proprietary sequence. AIS is not writable if AIE is high. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0 Array Integrity sequence is proprietary sequence. 1 Array Integrity sequence is sequential.
30 AIE	Array Integrity Enable Setting AIE to '1' starts the Array Integrity Check and the Margin Read on all selected blocks. The sequence is selected by AIS and the MISR (UM0–UM9) is checked on completion to determine if a correct signature is obtained. AIE can be set only if MCR.ERS, MCR.PGM and MCR.EHV are all low. AIE is not simultaneously writable to '1' while UTE is being cleared to '0'. Clearing AIE terminates the check when an Array Integrity operation finishes (AID=1) and aborts the check when and Array Integrity operation is ongoing (AID=0). AIE remains '1' when the Array Integrity Check - Margin read are suspended or are in breakpoint and cannot be cleared. (AIE can be cleared if NAIBP=0 and AISU=0) 0 Array Integrity Check and Margin Read and are not enabled. 1 Array Integrity Check and Margin Read are enabled.
31 AID	Array Integrity Done AID is set to indicate that the Array Integrity Check - Margin Read is complete and the MISR (UM0–9) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set) and the breakpoint condition is met. AID may also assert if Suspend is set (AISUS 0–1). AID can not be written, and is status only. 0 Array Integrity Check is on-going. 1 Array Integrity Check is done.

35.3.2.17 User Multiple Input Signature registers

The Multiple Input Signature registers allow evaluation of the Array Integrity or Margin Read run on a block-by-block basis by accumulating signatures from data bit fields.

These registers are not accessible whenever MCR.DONE or UT0.AID are low. Reading returns indeterminate data while writing has no effect.

The registers can be written in order to seed signatures before starting an AIC or MR operation.

Note: Do not write to the MISR registers during an Array Integrity operation (including Suspend or Breakpoint) as it may alter the final signature in an unpredictable way.

If reads extend into an invalid address location during address sequencing (that is greater than the maximum address for a given array size), reads are still executed to the array but the results are not deterministic: MISR registers are not recalculated and the previous value is retained.

35.3.2.17.1 User Multiple Input Signature 0 register (UM0)

UM0 represents bits 31–0 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0058 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MISR31	MISR30	MISR29	MISR28	MISR27	MISR26	MISR25	MISR24	MISR23	MISR22	MISR21	MISR20	MISR19	MISR18	MISR17	MISR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MISR15	MISR14	MISR13	MISR12	MISR11	MISR10	MISR9	MISR8	MISR7	MISR6	MISR5	MISR4	MISR3	MISR2	MISR1	MISR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 334. User Multiple Input Signature 0 register (UM0)

Table 429. UM0 field descriptions

Field	Description
0:31 MISR[31:0]	Multiple input signature 31–0 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.2 User Multiple Input Signature 1 register (UM1)

UM1 represents bits 63–32 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x005C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MISR63	MISR62	MISR61	MISR60	MISR59	MISR58	MISR57	MISR56	MISR55	MISR54	MISR53	MISR52	MISR51	MISR50	MISR49	MISR48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MISR47	MISR46	MISR45	MISR44	MISR43	MISR42	MISR41	MISR40	MISR39	MISR38	MISR37	MISR36	MISR35	MISR34	MISR33	MISR32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 335. User Multiple Input Signature 1 register (UM1)

Table 430. UM1 field descriptions

Field	Description
0:31 MISR[63:32]	Multiple input signature 63–32 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.3 User Multiple Input Signature 2 register (UM2)

UM2 represents bits 95–64 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0060 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MISR95	MISR94	MISR93	MISR92	MISR91	MISR90	MISR89	MISR88	MISR87	MISR86	MISR85	MISR84	MISR83	MISR82	MISR81	MISR80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MISR79	MISR78	MISR77	MISR76	MISR75	MISR74	MISR73	MISR72	MISR71	MISR70	MISR69	MISR68	MISR67	MISR66	MISR65	MISR64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 336. User Multiple Input Signature 2 register (UM2)

Table 431. UM2 field descriptions

Field	Description
0:31 MISR[95:64]	Multiple input signature 95–64 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.4 User Multiple Input Signature 3 register (UM3)

UM3 represents bits 127–96 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0064 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MISR127	MISR126	MISR125	MISR124	MISR123	MISR122	MISR121	MISR120	MISR119	MISR118	MISR117	MISR116	MISR115	MISR114	MISR113	MISR112
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MISR111	MISR110	MISR109	MISR108	MISR107	MISR106	MISR105	MISR104	MISR103	MISR102	MISR101	MISR100	MISR99	MISR98	MISR97	MISR96
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 337. User Multiple Input Signature 3 register (UM3)

Table 432. UM3 field descriptions

Field	Description
0:31 MISR[127:96]	Multiple input signature 127–96 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.5 User Multiple Input Signature 4 register (UM4)

UM4 represents bits 159–128 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0068 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MISR159	MISR158	MISR157	MISR156	MISR155	MISR154	MISR153	MISR152	MISR151	MISR150	MISR149	MISR148	MISR147	MISR146	MISR145	MISR144
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MISR143	MISR142	MISR141	MISR140	MISR139	MISR138	MISR137	MISR136	MISR135	MISR134	MISR133	MISR132	MISR131	MISR130	MISR129	MISR128
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 338. User Multiple Input Signature 4 register (UM4)

Table 433. UM4 field descriptions

Field	Description
0:31 MISR[159:128]	Multiple input signature 159–128 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.6 User Multiple Input Signature 5 register (UM5)

UM5 represents bits 191–160 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x006C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR191	MISR190	MISR189	MISR188	MISR187	MISR186	MISR185	MISR184	MISR183	MISR182	MISR181	MISR180	MISR179	MISR178	MISR177	MISR176
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR175	MISR174	MISR173	MISR172	MISR171	MISR170	MISR169	MISR168	MISR167	MISR166	MISR165	MISR164	MISR163	MISR162	MISR161	MISR160
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 339. User Multiple Input Signature 5 register (UM5)

Table 434. UM5 field descriptions

Field	Description
0:31 MISR[191:160]	Multiple input signature 191–160 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.7 User Multiple Input Signature 6 register (UM6)

UM6 represents bits 223–192 of the whole 288 bits word (4 doublewords including ECC).

Address: 0x0070 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR223	MISR222	MISR221	MISR220	MISR219	MISR218	MISR217	MISR216	MISR215	MISR214	MISR213	MISR212	MISR211	MISR210	MISR209	MISR208
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR207	MISR206	MISR205	MISR204	MISR203	MISR202	MISR201	MISR200	MISR199	MISR198	MISR197	MISR196	MISR195	MISR194	MISR193	MISR192
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 340. User Multiple Input Signature 6 register (UM6)

Table 435. UM6 field descriptions

Field	Description
0:31 MISR[223:192]	Multiple input signature 223–192 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.8 User Multiple Input Signature 7 register (UM7)

UM7 represents bits 255–224 of the whole 288 bits word (4 doublewords including ECC).

Address: 0x0074 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR255	MISR254	MISR253	MISR252	MISR251	MISR250	MISR249	MISR248	MISR247	MISR246	MISR245	MISR244	MISR243	MISR242	MISR241	MISR240
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR239	MISR238	MISR237	MISR236	MISR235	MISR234	MISR233	MISR232	MISR231	MISR230	MISR229	MISR228	MISR227	MISR226	MISR225	MISR224
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 341. User Multiple Input Signature 7 register (UM7)

Table 436. UM7 field descriptions

Field	Description
0:31 MISR[255:224]	Multiple input signature 255–224 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.9 User Multiple Input Signature 8 register (UM8)

UM8 represents the ECC parity bits of the whole 288-bits word (8 parity bits for each of the 4 doublewords).

Address: 0x0078 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MISR287	MISR286	MISR285	MISR284	MISR283	MISR282	MISR281	MISR280	MISR279	MISR278	MISR277	MISR276	MISR275	MISR274	MISR273	MISR272
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MISR271	MISR270	MISR269	MISR268	MISR267	MISR266	MISR265	MISR264	MISR263	MISR262	MISR261	MISR260	MISR259	MISR258	MISR257	MISR256
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 342. User Multiple Input Signature 8 register (UM8)

Table 437. UM8 field descriptions

Field	Description
0:31 MISR[287:256]	Multiple input signature 287–256 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.17.10 User Multiple Input Signature 9 register (UM9)

UM9 represents the ERR bits.

Address: 0x007C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR319	MISR318	MISR317	MISR316	MISR315	MISR314	MISR313	MISR312	MISR311	MISR310	MISR309	MISR308	MISR307	MISR306	MISR305	MISR304
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR303	MISR302	MISR301	MISR300	MISR299	MISR298	MISR297	MISR296	MISR295	MISR294	MISR293	MISR292	MISR291	MISR290	MISR289	MISR288
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 343. User Multiple Input Signature 9 register (UM9)

Table 438. UM9 field descriptions

Field	Description
0:31 MISR[319:288]	Multiple input signature 319–288 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

35.3.2.18 Over-program protection registers

Over-program protection prevents blocks from being over-programmed. These registers show the over-program protection status. See *Section 35.4.5.1.2: Over-programming protection (OPP) enable*.

35.3.2.18.1 Over-program protection 0 register (OPP0)

The Low/Mid Address Space is covered by the Over Program Protection 0 (OPP0) register.

Address: 0x0080 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWOPP													
W																
Reset[(1)]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDOPP															
W																
Reset[(1)]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 344. Over-program protection 0 register (OPP0)

Table 439. OPP0 field descriptions

Field	Description
2:15 LOWOPP	Low address space Over Program Protection The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWOPP field is not writable, and is status only. The default value of the LOWOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the LOWOPP bits default to not over-programming protected and will remain 0. 0 Low Address Space Block is unprotected from over-programming. 1 Low Address Space Block is protected from over-programming.
16:31 MIDOPP	Mid address space Over Program Protection The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDOPP field is not writable, and is status only. The default value of the MIDOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the MIDOPP bits default to not over-programming protected, and will remain 0. 0 Mid Address Space Block is unprotected from over-programming. 1 Mid Address Space Block is protected from over-programming.

35.3.2.18.2 Over-program protection 1 register (OPP1)

The High Address Space is covered by the Over Program Protection 1 (OPP1) register.

Address: 0x0084 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of “—” indicates that the value is set at the factory and can vary among different applications

Figure 345. Over-program protection 1 register (OPP1)

Table 440. OPP1 field descriptions

Field	Description
16:31 HIGHOPP	High address space Over Program Protection The block numbering for the High Blocks starts at HIGHOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHOPP field is not writable, and is status only. The default value of the HIGHOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the HIGHOPP bits default to not over-programming protected and will remain 0. 0 High Address Space Block is unprotected from over-programming. 1 High Address Space Block is protected from over-programming.

35.3.2.18.3 Over-program protection 2 register (OPP2)

The A256K Address Space Block is covered by the Over Program Protection 2 (OPP2) register.

Address: 0x0088 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 346. Over-program protection 2 register (OPP2)

Table 441. OPP2 field descriptions

Field	Description
0:31 A256KOPP	256K Address Space Block Over Program Protection A256KOPP[31:0] The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted for. The A256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A256KOPP bits default to not over-programming protected and will remain 0. 0 A256K Address Space Block is unprotected from over-programming. 1 A256K Address Space Block is protected from over-programming.

35.3.2.18.4 Over-program protection 3 register (OPP3)

The 256K high Address Space is covered by the Over Program Protection 3 (OPP3) register.

Address: 0x008C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 347. Over-program protection 3 register (OPP3)

Table 442. OPP3 field descriptions

Field	Description
0:31 A256KOPP	256K Address space Over Program Protection A256KOPP[63:32] The block numbering for the 256 KB Blocks starts at A256KOPP[32] and continues until all blocks are accounted for. The A256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A256KOPP bits default to not over-programming protected and will remain 0. 0 A256K Address Space Block is unprotected from over-programming. 1 A256K Address Space Block is protected from over-programming.

35.3.2.19 Test Mode Disable Password Check register (TMD)

The Test Mode Disable Password Check (TMD) register provides a means of entering the password to be matched with the one stored in UTEST location in order to allow manufacturer access to protected Test Mode blocks by writing 0x5A4B_3C2D in the Test Mode Disable Seal location.

Address: 0x0090 Access: User Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 348. Test Mode Disable Password Check register (TMD)

Table 443. TMD field descriptions

Field	Description
0:31 PWD	The Test Mode Disable Check register is used as a location to provide a password challenge to unlock blocks selected to be sealed as part of the Test Mode Disable seal mechanism. Please see *Section 35.4.8.2: Test Mode Disable* for more information. Writes to the register have no effect except for a password challenge. Reads to this register will always return 0. The register's reset value is 0. **Note:** PWD bit field is byte-reversed compared to UTEST TMD password: for instance, if UTEST TMD password is written 0x01020304, the PWD bit field must be written 0x04030201

35.3.2.20 HSM Alternate selector registers

HSM Alternate selectors assign blocks from default to ALTERNATE Program & Erase interface. These registers show the HSM Alternate assignment status. See *Section 35.4.6: Alternate program and erase interface*.

35.3.2.20.1 HSM ALT-SEL register 0 (ALTSEL0)

The Low/Mid Address Space is assigned between MAIN and ALT Interface. Through the HSM ALT-SEL register it is possible to read the actual status of the assignment.

Address: 0x0094 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWALTSEL													
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDALTSEL															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of “—” indicates that the value is set at the factory and can vary among different applications

Figure 349. HSM ALT-SEL register 0 (ALTSEL0)

Table 444. ALTSEL0 field descriptions

Field	Description
2:15 LOWALTSEL	Low address space HSM Alternate Select The block numbering for the Low Blocks starts at LOWALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWALTSEL bits will remain 0. 0 Low Address Space Block is assigned to Main interface. 1 Low Address Space Block is assigned to Alternate interface.
16:31 MIDALTSEL	Mid address space HSM Alternate Select 0 The block numbering for the Mid Blocks starts at MIDALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDALTSEL bits will remain 0. 0 Mid Address Space Block is assigned to Main interface. 1 Mid Address Space Block is assigned to Alternate interface.

35.3.2.20.2 HSM ALT-SEL register 1 (ALTSEL1)

The High Address Space is assigned between MAIN and ALT Interface. Through the HSM ALT-SEL register it is possible to read the actual status of the assignment.

Address: 0x0098 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHALTSEL															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 350. HSM ALT-SEL register 1 (ALTSEL1)

Table 445. ALTSEL1 field descriptions

Field	Description
16:31 HIGHALTSEL	High address space HSM Alternate Select The block numbering for the High Blocks starts at HIGHALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHALTSEL bits will remain 0. 0 High Address Space Block is assigned to Main interface. 1 High Address Space Block is assigned to Alternate interface.

35.3.2.21 Erase Lock protection registers

Erase Lock protection prevents blocks from being erase. These registers show the Erase Lock protection status. See *Section 35.4.8.1: Modify Protection*.

35.3.2.21.1 Erase Lock Protection 0 register (ELOCK0)

The Low/Mid Address Space is covered by the Erase Lock Protection 0 (ELOCK0) register.

Address: 0x0100 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LOWELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 351. Erase Lock Protection 0 register (ELOCK0)

Table 446. ELOCK0 field descriptions

Field	Description
0:15 LOWELOCK	Low address space Erase Lock Protection The block numbering for the Low Blocks starts at LOWELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWELOCK bits default is Erase Lock protected and will remain 1. 0 Low Address Space Block is Erase unprotected. 1 Low Address Space Block is Erase protected.
16:31 MIDELOCK	Mid address space Erase Lock Protection The block numbering for the Mid Blocks starts at MIDELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDELOCK bits default is Erase Lock protected and will remain 1. 0 Mid Address Space Block is Erase unprotected. 1 Mid Address Space Block is Erase protected.

35.3.2.21.2 Erase Lock Protection 1 register (ELOCK1)

The High Address Space is covered by the Erase Lock Protection 1 (ELOCK1) register.

Address: 0x0104 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 352. Erase Lock Protection 1 register (ELOCK1)

Table 447. ELOCK1 field descriptions

Field	Description
16:31 HIGHELOCK	High address space Erase Lock Protection The block numbering for the High Blocks starts at HIGHELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHELOCK bits default is Erase Lock protected and will remain 1. 0 High Address Space Block is Erase unprotected. 1 High Address Space Block is Erase protected.

35.3.2.21.3 Erase Lock Protection 2 register (ELOCK2)

The 256K Address Space is covered by the Erase Lock Protection 2 (ELOCK2) register.

Address: 0x0108 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A256KELOCK															
W																
Reset[(1)]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KELOCK															
W																
Reset[(1)]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of “—” indicates that the value is set at the factory and can vary among different applications

Figure 353. Erase Lock Protection 2 register (ELOCK2)

Table 448. ELOCK2 field descriptions

Field	Description
0:31 A256KELOCK	256K High address space Erase Lock Protection A256KELOCK[31:0] The block numbering for the 256 KB Blocks starts at A256KELOCK[0] and continues until all blocks are accounted for. The A256KELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KELOCK bits default is Erase Lock protected and will remain 1. 0 A256K Address Space Block is Erase unprotected. 1 A256K Address Space Block is Erase protected.

35.3.2.21.4 Erase Lock protection 3 register (ELOCK3)

The 256K Address Space is covered by the Erase Lock Protection 3 (ELOCK3) register.

Address: 0x010C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A256KELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 354. Erase Lock protection 3 register (ELOCK3)

Table 449. ELOCK3 field descriptions

Field	Description
0:31 A256KELOCK	256K High address space Erase Lock Protection A256KELOCK[63:32] The block numbering for the 256 KB Blocks starts at A256KELOCK[32] and continues until all blocks are accounted for. The A256KELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KELOCK bits default is Erase Lock protected and will remain 1. 0 A256K Address Space Block is erase unprotected. 1 A256K Address Space Block is Erase protected.

35.3.2.22 Program Lock protection registers

Program Lock protection prevents blocks from being programmed. These registers show the Program Lock protection status. See *Section 35.4.8.1: Modify Protection*.

35.3.2.22.1 Program Lock Protection 0 register (PLOCK0)

The Low/Mid Address Space is covered by the Program Lock Protection 0 (PLOCK0) register.

Address: 0x0110 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LOWPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 355. Program Lock Protection 0 register (PLOCK0)

Table 450. PLOCK0 field descriptions

Field	Description
0:15 LOWPLOCK	Low address space Erase Lock Protection The block numbering for the Low Blocks starts at LOWPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWPLOCK bits default is Program Lock protected and will remain 1. 0 Low Address Space Block is Program unprotected. 1 Low Address Space Block is Program protected.
16:31 MIDPLOCK	Mid address space Erase Lock Protection The block numbering for the Mid Blocks starts at MIDPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDPLOCK bits default is Program Lock protected and will remain 1. 0 Mid Address Space Block is Program unprotected. 1 Mid Address Space Block is Program protected.

35.3.2.22.2 Program Lock Protection 1 register (PLOCK1)

The High Address Space is covered by the Program Lock Protection 1 (PLOCK1) register.

Address: 0x0114 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of “—” indicates that the value is set at the factory and can vary among different applications

Figure 356. Erase Lock Protection 1 register (PLOCK1)

Table 451. PLOCK1 field descriptions

Field	Description
16:31 HIGHPLOCK	High address space Program Lock Protection The block numbering for the High Blocks starts at HIGHPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHPLOCK bits default is Program Lock protected and will remain 1. 0 High Address Space Block is Program unprotected. 1 High Address Space Block is Program protected.

35.3.2.22.3 Program Lock Protection 2 register (PLOCK2)

The 256K Address Space is covered by the Program Lock Protection 2 (PLOCK2) register.

Address: 0x0118 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A256KPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of “—” indicates that the value is set at the factory and can vary among different applications

Figure 357. Program Lock Protection 2 register (PLOCK2)

Table 452. PLOCK2 field descriptions

Field	Description
0:31 A256KPLOCK	256K High address space Program Lock Protection A256KPLOCK[31:0] The block numbering for the 256 KB Blocks starts at A256KPLOCK[0] and continues until all blocks are accounted for. The A256KPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KPLOCK bits default is Program Lock protected and will remain 1. 0 A256K Address Space Block is Program unprotected. 1 A256K Address Space Block is Program protected.

35.3.2.22.4 Program Lock protection 3 register (PLOCK3)

The 256K Address Space is covered by the Program Lock Protection 3 (PLOCK3) register.

Address: 0x011C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A256KPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 358. Program Lock protection 3 register (PLOCK3)

Table 453. PLOCK3 field descriptions

Field	Description
0:31 A256KPLOCK	256K High address space Program Lock Protection A256KPLOCK[63:32] The block numbering for the 256 KB Blocks starts at A256KPLOCK[32] and continues until all blocks are accounted for. The A256KPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KPLOCK bits default is Program Lock protected and will remain 1. 0 A256K Address Space Block is Program unprotected. 1 A256K Address Space Block is Program protected.

For the address mapping, refer to *Section 35.3.2.15: Address register (ADR)* description.

35.4 Functional description

This chapter describes the details of all the modes of operation available to the Flash Memory module.

35.4.1 Reset

A reset is the highest priority operation for the Flash Memory module and terminates all other operations.

The Flash Memory module uses reset to initialize register and status bits to their default reset values. If the Flash Memory module is executing a Program or Erase operation (MCR.PGM=1 or MCR.ERS=1) and a reset is issued, the operation is immediately terminated and the module disables the high voltage circuits.Reset terminates all operations and forces the Flash Memory module into Read mode ready to receive Read/Write accesses.

Warning: Reset and Power-Off must not be used as a systematic way to terminate a Program or Erase operation.

After reset is negated Read register access may be performed, although registers that require updating from shadow information or other inputs may not read updated values until MCR.DONE transforms. MCR.DONE may be polled to determine if the Flash Memory module has transitioned out of reset. Note that the registers cannot be written until MCR.DONE is high.

35.4.2 Power-down mode (Disable mode)

The Power-down (or Disable) mode allows the turning-off of all Flash DC current sources. All power dissipation in this mode is due solely to leakage.

In Power-down mode no Read from or Write to the Flash array is possible. Every Flash array Read access executed when the Power-down mode is active outputs invalid data,.

The user may not read some registers (UM0–UM9 and part of UT0) until the Power-down mode is exited.On the contrary, Write access is locked on all the registers in Power-Down mode.

When enabled the Flash Memory module returns to its predisable state in all cases unless in the process of executing an Erase or a Program High Voltage operation at the time of disable. If the Flash Memory module is disabled during an Erase operation, the operation is suspended and MCR.ESUS bits are set to ‘1’. The user may resume the Erase operation at the time the module is enabled by clearing MCR.ESUS bit. MCR.EHV must be high to resume the Erase operation.

If the Flash Memory module is disabled during a program operation, the operation is suspended and MCR.PSUS bit is set to ‘1’. The user may resume the program operation at the time the module is enabled by clearing MCR.PSUS bit. MCR.EHV must be high to resume the Erase operation.

If the Flash Memory module is disabled during an erase-suspended program operation, the program operation is suspended and both ESUS and PSUS are set to a ‘1’. When the module is enabled PSUS must be cleared first to resume the program. Erase may be resumed after the program ends. EHV must be high to resume both operations.

If the Flash Memory module is disabled during AIC or MR, the operation is aborted and the MISR value is not deterministic.

If the Flash Memory module is put in Power-down mode and the Vector Table remains mapped in the Flash Address space, the user must be mindful that the Flash Memory module significantly increases the interrupt response time by adding several Wait States.

Note: Do not enter Low-power mode when the Power-down mode is active.

35.4.3 Low-power mode (Sleep mode)

The Low-power (or Sleep) mode turns-off most of the DC current sources within the Flash Memory module. Wake-up time from Low-power mode is faster than wake-up time from Power-down mode.

In Low-power mode no Read from or Write to the Flash array is possible. Every Read access executed when the Low-power mode is active outputs invalid data.

The user may not read some registers (UM0–UM9 and part of UT0) until the Low-power mode is exited. Write access is locked on all the registers in Low-power mode.

When exiting from Low-power mode the Flash Memory module returns to its presleep state in all cases unless in the process of executing an Erase High Voltage operation at the time of entering sleep mode. If the Flash Memory module is put in sleep mode during an Erase operation, MCR.ESUS is set to ‘1’. The user may resume the Erase operation at the time the module exits from sleep by clearing MCR.ESUS. MCR.EHV must be high to resume the Erase operation.

If the Flash Memory module is put in Low-power mode during a program operation, the operation is suspended and MCR.PSUS is set to ‘1’. The user may resume the program operation at the time the module is enabled by clearing MCR.PSUS. MCR.EHV must be high to resume the Erase operation.

If the Flash Memory module is put in Low-power mode during an erase-suspended program operation, the program operation is suspended and both ESUS and PSUS are set to a ‘1’. When the module is enabled, PSUS must first be cleared for the program to resume. The Erase may be resumed after the program ends. EHV must be high to resume both operations.

If the Flash Memory module is disabled during AIC or MR, the operation is aborted and the MISR value is not deterministic.

Note: Do not enter Power-down mode when the Low-power mode is active.

35.4.4 Read mode

Read mode is the active mode of operation of the Flash Memory module in which the Flash array can be read or ‘written’ (the so-called Interlock Write), or the Flash registers can be read or written.

The Flash Memory module enters Read mode when exiting from reset. The default state of the Flash Memory module is Read mode.

The main spaces (Low, Mid, High, and 256K) and UTEST address space of Flash array can only be read in Read mode.

Flash registers can even be read when the module is in Power-down or Low-power mode (except a few documented registers) as well as in Read mode. However, Flash registers can only be written while in Read mode.

Flash memory module is in Read mode, granting the same access time, under the following conditions:

- The Read state is active when the module is enabled (User Mode Read)
- The Read state is active when MCR.ERS and MCR.ESUS are high and MCR.PGM is low (Erase Suspend)
- The Read state is active when MCR.PGM and MCR.PSUS are high (Program Suspend)
- The Read state is active when MCR.PGM or MCR.ERS are high and High Voltage operation is ongoing on one partition, thus MCR.EHV is high (condition referred as Read-While-Write). In this case reads done on the other partitions are allowed, while reads done to the partition being operated on (either Erase or Program) result in an

error with MCR.RWE set and the data read back possibly being random including the possibility of ECC errors.

- The Read state is active when MCRA.ERSA and MCRA.ESUSA are high and MCRA.PGMA is low (Alternate Erase Suspend)
- The Read state is active when MCRA.PGMA and MCRA.PSUSA are high (Alternate Program Suspend)
- The Read state is active when MCRA.PGMA or MCRA.ERSA are high and High Voltage operation is ongoing on one partition, thus MCRA.EHVa is high (condition referred as Alternate Read-While-Write). In this case reads done on the other partitions are allowed, while reads done to the partition being operated on (either Alternate Erase or Alternate Program) result in an error with MCRA.RWEA set and the data read back possibly being random including the possibility of ECC errors.

Code Flash partitions reads return 256 bits (1 page = 4 doublewords). Data Flash partition reads return 128 bits. Register reads return 32 bits (1 word).

Flash array reads are performed through the Bus Interface Unit. In many cases the BIU applies “Read Page Buffering” to allow sequential reads to be executed with higher performance. This can provide data coherency issues that must be handled by software. Data coherency may be an issue after a program or an Erase operation, as well as Test block operations.

Flash array reads to invalid locations result in indeterminate data. Invalid locations occur when addressing points to blocks that are not included in the Flash memory Map of any partition. Interlock writes to invalid locations result in an interlock occurring, but attempts to program these blocks do not eventuate since they are forced into a locked state. Erase occurs to selected and unlocked blocks even if the Interlock Write is to an invalid location.

Register reads to unmapped register address spaces return all 0’s. Registers writes to unmapped register address spaces have no effect.

Simultaneous Read cycles on the Flash array and Read/Write cycles on the registers are possible, but not simultaneous register Read/Write accesses with Flash Array Interlock Writes are forbidden.

35.4.4.1 Error correction code

The Flash Memory module implements a strategy for improving the reliability of the data stored in Flash via the use of Error Correction Code.
In Code Memory the ECC size is 128 bits with 17 ECC bits associated. In Data Memory the ECC size is 64 bits with 15 ECC bits associated.
ECC function is implemented in such a way to guarantee Double Error Correction and Triple Error Detection (DEC-TED) in Data Memory.
ECC function is implemented in such a way to guarantee Double Error Correction and Triple Error Detection (DEC-TED) in Code Memory.
ECC coding has all 0’s (intended as data and parity) as an invalid word and all 1’s as a valid word.

In addition to providing the requested Read data, the Flash also provides output sidebands reflecting the status of the single and double error corrections and triple error detections performed at the end of each Read.

For functional safety coverage, Flash performs an EDC post ECC operation at every Read to detect mismatches in the ECC logic detection and correction. The result is output as sideband. Also active is a monitor circuit that checks the reference current in the sense amplifiers, including control of the internally regulated Read voltage, to prevent anomalous Read operation.

ECC circuitry provides single-bit fault correction used to achieve automotive reliability targets. Some units experience single-bit corrections throughout the life of the product with no impact on product reliability.

35.4.4.1.1 Multi-bits error management (more errors than uncorrectable detection capability)

In this case, behavior cannot be predicted without knowing the original data and the position of the bit flips. The hardware can react in multiple ways:

1. do nothing (that is, not signaling of any flag)
2. assert either SBC, SBC1, or EER without EEE
3. assert either SBC, SBC1, or EER with EEE
4. assert EEE only.

Note: SBC and SBC1 reporting have to be activated in UT0.

It is worth highlighting that by design the purpose of the ECC logic is not to detect multi-bit errors but cope with errors in the data and parity containing maximum number of bit flip till uncorrectable detection capability. In this condition, the EDC after ECC monitors the integrity of the ECC logic.

Other mechanisms must detect multi-bit errors or avoid their occurrence.

For example running the Array Integrity Check once after the boot before the safety application starts to protect the Data with an application level checksum.

An example of a mechanism that decreases the possibility of multi-bit errors in the array is the multiplexing. The bits of the same word are not one next to each other, but they are spaced of a fixed number of bits.

35.4.4.1.2 EEPROM emulation

The chosen ECC allows some bit manipulation so that a doubleword can be re-written several times without needing an erase of the block. This allows the use of a doubleword to store flags useful for the EEPROM emulation.

The sequence provided in *Table 454: Bits Manipulation: doublewords preserving ECC integrity* is related to Data Memory (therefore would not apply on Code Memory). The ECC coding allows, starting from an all 1's doubleword value, to rewrite subsequent predefined code values changing only 0s into 1s and preserving the ECC integrity.

Caution: Predefined code values have to be applied following the exact Byte order, as reported in the table.

Table 454. Bits Manipulation: doublewords preserving ECC integrity

Doubleword	ECC parity bits	Sequence
B7-B6-B5-B4-B3-B2_B1-B0		
0xFFFF_FFFF_FFFF_FFFF	0x7FFF	erase
0xFFFF_FFFF_FFFF_FFFE	0x3C88	write1
0xFFFF_FFFF_FEFE_FC80	0x3C88	write2

Table 454. Bits Manipulation: doublewords preserving ECC integrity (continued)

Doubleword	ECC parity bits	Sequence
B7-B6-B5-B4-B3-B2_B1-B0		
0xFFFF_FFFC_FEFC_FC80	0x1880	write3
0xFCFC_FFFC_FCFC_F880	0x1880	write4
0xFCF8_F8F0_FCE0_E080	0x1800	write5
0xF8F8_E080_F0E0_E000	0x1000	write6
0xE0E0_E080_E0E0_C000	0x1000	write7
0xE000_8000_00C0_0000	0x1000	write8

35.4.4.2 Flash address generation check

For functional safety coverage, Flash provides output sidebands reflecting the re-encoded address used to perform the actual Read operation. In the re-encoding process both word line and column within the selected block are evaluated. For the completion of the re-encoding operation, the addresses related to block selections are also computed. Re-encoding circuitry is able to detect both no-selection as well as multi-selection. Flash provides a self-check of address consistency comparing at each Read the sampling of the current Read address with the re-encoded one, and in case of mismatch an error sideband is set.

35.4.5 Modify mode

All the Modify operations of the Flash Memory module are managed through the Flash User Registers interface. This interface is referred as MAIN.

When a Modify operation is active on some blocks, no Read access is possible on any other block of the same Flash Partition. The Read-While-Modify is supported only among different Flash partitions.

During a Flash Modify operation any attempt to read any Flash location in the same Flash partition outputs invalid data and MCR.RWE is automatically set.

If a reset occurs during a Modify operation, the operation is immediately terminated and the module is reset to Read mode. The data integrity of the Flash section where the Modify operation was terminated is not guaranteed: the interrupted Flash Modify operation must be repeated.

In general each Modify operation is started through a sequence of 3 steps:

1. The first instruction is used to select the desired operation by setting its corresponding selection bit in MCR.PGM or MCR.ERS or UT0.MRE.
2. The second step is the definition of the operands: the address and the data for Programming, or the blocks for Array Integrity Check or Margin Read.
3. The third instruction is used to start the Modify operation by setting MCR.EHV or UT0.AIE.

Once selected, but not yet initiated, one operation can be canceled by resetting the operation selection bit.

A summary of the available Flash Modify operations are shown in *Table 455*.

Table 455. Flash Modify operations

Operation	Select bit	Operands	Start bit
Doubleword Program	MCR.PGM	Address and Data by Interlock Writes	MCR.EHV
Block Erase	MCR.ERS	SEL0, SEL1, SEL2 and SEL3	MCR.EHV
Array Integrity Check	None	SEL0, SEL1, SEL2 and SEL3	UT0.AIE
Margin Read	UT0.MRE	UT0.MRV + SEL0, SEL1, SEL2 and SEL3	UT0.AIE

Once MCR.EHV (or UT0.AIE) is set, all the operands can no longer be modified until MCR.DONE (or UT0.AID) is high.

In general each Modify operation is completed through a sequence of 4 steps:

1. Wait for operation completion: wait for MCR.DONE (or UT0.AID) to go high.
2. Check operation result: check bit MCR.PEG (or compare UM0–UM9 with expected value).
3. Switch-Off FPEC by resetting MCR.EHV (or UT0.AIE).
4. Deselect current operation by clearing MCR.PGM/ERS (or UT0.MRE).

Note: Do not initiate a Modify operation on a Flash Memory module while another Modify operation is in progress on a separate module.

In the following sections, all the possible Modify operations are described with accompanying examples of the sequences needed to activate them.

35.4.5.1 Program

A Flash Program sequence operates on any doubleword within the Flash core. Up to two words within the doubleword may be altered in a single doubleword program operation. A single program operation can also alter one page (4 doublewords, 32 bytes), or 4 pages max (128 bytes in total). It is possible to program only certain doublewords in the page or the quad-page, leaving the others with their original content. Whenever the array is programmed, the ECC bits are also programmed.

ECC is handled on a 64-bit boundary in Data Flash and 128b boundary in Code Flash. Whenever you program, ECC bits are also programmed. Thus, if only one word in any given ECC segment is programmed, the adjoining word/s (in that segment) should not be programmed later on, since ECC calculation has already completed for that segment. Attempts to program the adjoining word will probably result in an operation failure. All programming operations should be from 64 bits to 1024 bits, should be 64-bit aligned for Data Flash and 128-bit aligned for Code Flash. The programming operation should completely fill selected ECC segments.

Due to the fact that Flash functions in a System with an end2end ECC scheme, ECC parity bit are also input during an Interlock Write. After setting MCR.EHV, the FPEC checks the integrity of the data received in terms of ECC. If any doubleword in the doubleword program, the page program or the quad-page program contains an inconsistency between data and parity, the whole operation is rejected and a MCR.PEG=0 is set. The doublewords without data and with parity inconsistency are thus not programmed.

Programming changes the value stored in an array bit from logic ‘1’ to logic ‘0’ only. Programming cannot change a stored logic ‘0’ to a logic ‘1’. If a logic ‘0’ is attempted to be over programmed by a logic ‘1’, the resulting operation fails (MCR.PEG=0), and the 0s that are interlocked are merged (ORed) with 0s that are already present in the 64-bit ECC segment. Extreme care has to be taken when doing such an operation due to the high risk of producing ECC errors.

Addresses in locked blocks cannot be programmed. Register locked blocks cannot be programmed. Hardware locking is also available which only affects the program operation. Hardware locking is enabled by sideband signals, and if enabled, programs are prevented from accessing hardware locked blocks in the same way as program software locking.

The user may program the values in any or all of eight the words in a page with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. Only certain doublewords may be programmed within the page or the quad-page, without altering the content of the ones unselected via interlock.

The Program operation consists of the following sequence of events:

1. Change the value of MCR.PGM from ‘0’ to ‘1’.
2. Ensure the block that contains the address to be programmed is unlocked. Write the first address to be programmed with the program data. The Flash Memory module latches address bits (23:7) at this time, as well as written data. This Write is referred to as a Program Data Interlock Write. An Interlock Write may be as large as 64 bits, and as small as 32 bits (depending on the CPU bus).
3. If more than 1 word is to be programmed, write the additional address in the doubleword with data to be programmed. This is referred to as a Program Data Write. The Flash Memory modules ignore address bits (23:7) for program data writes. The eventual unwritten data word defaults to 0xFFFFFFFF.
4. Write a logic ‘1’ to the MCR.EHV to start the internal program sequence or terminate via step 9.
5. Wait until the MCR.DONE goes high.
6. Confirm MCR.PEG=1.
7. Write a logic ‘0’ to the MCR.EHV.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic ‘0’ to the MCR.PGM to terminate the program operation.

Programs may be initiated with the 0–1 transition of MCR.PGM or by clearing MCR.EHV at the end of a previous program. The first Write after a program is initiated determines the page address to be programmed. This first Write is referred to as an Interlock Write. The Interlock Write determines if the UTEST or normal array space will be programmed via MCR.PEAS.

In the case of an erase-suspended program, the values in MCR.PEAS may be modified via the Program Interlock Write, enabling erase-suspended programs to UTEST Nonvolatile Memory space and revert back to their original state (related to erase-suspended operation) once the erase-suspended program is completed.

An Interlock Write must be performed before setting MCR.EHV. An Interlock Write performed after setting MCR.EHV is forbidden and data and address passed through interlock is lost.The user may terminate a program sequence by clearing MCR.PGM prior to setting MCR.EHV.

After the Interlock Write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024 bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of

0xFFFF_FFFF. If multiple writes are performed on the same location, the data for the last Write is used in programming.

While MCR.DONE is low and MCR.EHV is high, the user may clear EHV, resulting in a Program Abort.

A Program Abort forces the module to step 8 of the program sequence.

An aborted program results in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The data space being operated on before the Abort contains indeterminate data. This may be recovered by repeating the same program instruction or executing an Erase of the affected blocks.

The user may not abort a program sequence while in Program Suspend.

Figure 359. Doubleword program of data 0x55AA_55AA at address 0x00_AAA8 and data 0xAA55_AA55 at address 0x00_AAAC

```
MCR = 0x0000_0010;             /* Set PGM in MCR: Select Operation */
(0x00_AAA8) = 0x55AA_55AA;     /* Latch Address and 32 LSB data */
(0x00_AAAC) = 0xAA55_AA55;     /* Latch 32 MSB data */
MCR = 0x0000_0011;             /* Set EHV in MCR: Operation Start */
do                             /* Loop to wait for DONE=1 */
{ tmp = MCR;                   /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
status = MCR & 0x0000_0200;    /* Check PEG flag */
MCR = 0x0000_0010;             /* Reset EHV in MCR: Operation End */
MCR = 0x0000_0000;             /* Reset PGM: Deselect Operation */
```

35.4.5.1.1 Program Suspend/Resume

The program sequence may be suspended to allow Read access to the Flash array. It is not possible to program or to erase during a Program Suspend. Read-While-Write operations may also be used to read the array during a program sequence providing the Read is to a different partition.

During Program Suspend, all reads to the doubleword targeted for program return indeterminate data.

A Program Suspend can be initiated by changing the value of MCR.PSUS from ‘0’ to ‘1’. MCR.PSUS can be set to ‘1’ at any time when MCR.PGM and MCR.EHV are high and MCR.ERS is low. 0–1 transition of MCR.PSUS causes the module to start the sequence which places it in Program Suspend.

It is possible to suspend a program performed during an Erase Suspend. The user must wait until MCR.DONE=1 before the module is suspended and further actions are attempted. MCR.DONE takes no more time than t_{PSUS} to go high after MCR.PSUS is set to ‘1’.

Once suspended, the array may be read. Flash memory reads while MCR.PSUS=1 from the doubleword being programmed return indeterminate data.

Figure 360. Doubleword Program Suspend

```
MCR = 0x0000_0019; /* Set PSUS in MCR: Program Suspend */
do                 /* Loop to wait for DONE=1 */
{ tmp = MCR;       /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
```

Note that there is no need to clear MCR.EHV and MCR.PGM in order to perform reads during program suspend.

The Program sequence is resumed by writing a logic ‘0’ to MCR.PSUS.

MCR.EHV must be set to ‘1’ before MCR.PSUS can be cleared to resume the operation.

The module continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

Figure 361. Doubleword Program Resume

```
MCR = 0x0000_0011;/* Reset PSUS in MCR: Program Resume */
```

Repeated suspends at a high frequency may result in the operation timing out, and the embedded Flash memory responds by completing the operation with a fail code (MCR.PEG=0). The minimum time between Program Suspends to ensure this does not occur is t_{PSRT}.

35.4.5.1.2 Over-programming protection (OPP) enable

Over Programming Protection can be enabled on a block-by-block basis. In an OPP enabled block of code Flash, any quad-word which has already been programmed cannot be programmed again. The OPP enable does not affect the Erase operation. It is possible to independently set OPP and (hardware) erase protection in order to avoid any kind of overwrite. Using these protection mechanisms in conjunction with the Test mode disable feature gives even greater overwrite protection. Attempts to over-program result in MCR.PEG being cleared.

OPP can be enabled for 16 KB, 32 KB, 64 KB and 256 KB blocks.

OPP is enabled by sideband signals and, if enabled, the program operation is modified to check for programmed bits prior to undertaking a high voltage program event.

35.4.5.1.3 Successful program address

To enable the System to create logic to authenticate operations with a program (like in a diary operation), the Flash Memory module provides the last successful address programmed as a sideband signal. This value is held until the next successful program. A successful program is defined as a program operation of data that is not all 1s, to an unlocked block and the operation results in MCR.PEG returning a ‘1’. An all 1’s value is the default (and invalid) address after reset.

In case of page programming, the last successful address returned is the beginning of the page itself regardless of whether that specific address has not been passed through interlock. In case of quad-page programming, the last successful address returned is the beginning of the quad-page itself regardless of whether that specific address has not been passed through interlock.

35.4.5.2 Erase

Erase changes the value stored in all bits of the selected block(s) to logic ‘1’. An Erase sequence operates on any combination of blocks in the low, mid, high or 256K address space. The Test block cannot be erased.

The Erase sequence is fully automated within the Flash. The user only needs to select the blocks to be erased and initiate the Erase sequence.

Register locked blocks cannot be erased. Hardware locking, which only affects the Erase operation is also available. Hardware locking is enabled by sideband signals and, if enabled, erases are prohibited on hardware locked blocks in the same way as erase software locking. If multiple blocks are selected for erase during an Erase sequence, no specific operation order can be assumed.

The Erase operation consists of the following sequence of events:

1. Change the value in the MCR.ERS bit from ‘0’ to ‘1’.
2. Select the block(s) to be erased by writing 1’s to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers. Note that Lock and Select are independent. If a block is selected and locked, no erase occurs.
3. Write to any address in Flash. This is referred to as an Erase Interlock Write.
4. Write a logic ‘1’ to MCR.EHV to start the internal Erase sequence or skip to step 9 to terminate.
5. Wait until MCR.DONE goes high.
6. Confirm MCR.PEG=1.
7. Write a logic ‘0’ to MCR.EHV.
8. If more blocks are to be erased, return to step 2.
9. Write a logic ‘0’ to the MCR.ERS bit to terminate the Erase operation.

After setting MCR.ERS, one Write (referred to as an Interlock Write) must be performed before MCR.EHV can be set to ‘1’.

Data words written during Erase sequence Interlock Writes are ignored. As Flash functions in an System with an end2end ECC scheme, ECC parity bits are also input during an Interlock Write. After setting MCR.EHV, the FPEC checks the integrity of the data received in terms of ECC even if the value of the data is not meaningful for the Erase operation. If an inconsistency between data and parity is found, the whole operation is rejected and MCR.PEG=0 is set.

The user may terminate the Erase sequence by clearing ERS before setting EHV.

An Erase operation may be aborted by clearing MCR.EHV assuming MCR.DONE is low, MCR.EHV is high and MCR.ESUS is low. An Erase Abort forces the module to step 8 of the Erase sequence. An aborted Erase results in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The block(s) being operated on before the Abort contain indeterminate data. This may be recovered by executing an Erase on the affected blocks. The user may not abort an Erase sequence while in Erase-Suspend.

Figure 362. Erase of blocks (example)

```
MCR = 0x0000_0004;            /* Set ERS in MCR: Select Operation */
SEL0 = 0x0006_0000;           /* Set the corresponding bit in SEL0: Select
Blocks */
(0x00_0000) = 0xFFFF_FFFF;    /* Latch a Flash Address */
MCR = 0x0000_0005;            /* Set EHV in MCR: Operation Start */
do                            /* Loop to wait for DONE=1 */
{ tmp = MCR;                  /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
status = MCR & 0x0000_0200;   /* Check PEG flag */
MCR = 0x0000_0004;            /* Reset EHV in MCR: Operation End */
```

35.4.5.2.1 Erase Suspend/Resume

The Erase sequence may be suspended to allow Read access to the Flash array. It is possible to program another block in any Flash Memory module during Erase-Suspend. Programming a location subject to suspended Erase results in the program not being executed and PEG being set to 0.

It is not possible to erase any block during an Erase-Suspend.

During Erase-Suspend, all reads to blocks targeted for Erase return indeterminate data.

An Erase-Suspend can be initiated by changing the value of MCR.ESUS from ‘0’ to ‘1’. MCR.ESUS can be set to 1 at any time when MCR.ERS and MCR.EHV are high and MCR.PGM is low. A 0–1 transition of MCR.ESUS causes the module to start the sequence which places it in Erase-Suspend. The User must wait until MCR.DONE=1 before the module is suspended and further actions are attempted.

MCR.DONE takes no longer than t_{ESUS} to go high after MCR.ESUS is set to ‘1’.

Once suspended, it is not possible to alter the content of SEL0, SEL1, SEL2 and SEL3 registers. Once suspended, the array may be read. Flash memory reads while MCR.ESUS=1 from the block(s) being erased return indeterminate data.

Figure 363. Block Erase Suspend

```
MCR = 0x0000_0007; /* Set ESUS in MCR: Erase Suspend */
do                 /* Loop to wait for DONE=1 */
{ tmp = MCR;       /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
```

Note that there is no need to clear MCR.EHV and MCR.ERS in order to perform reads during Erase-Suspend.

The Erase sequence is resumed by writing a logic ‘0’ to MCR.ESUS.

MCR.EHV must be set to ‘1’ before MCR.ESUS can be cleared to resume the operation. The module continues the Erase sequence from one of a set of predefined points and this may extend the time required for the Erase operation.

Figure 364. Block Erase Resume

```
MCR= 0x0000_0005;/* Reset ESUS in MCR: Erase Resume */
```

Repeated suspends at a high frequency may result in the operation timing out and the embedded Flash memory responds by completing the operation with a fail code (MCR.PEG=0). The minimum time between Erase suspends to ensure this does not occur is t_{ESRT}.

35.4.5.3 Factory Mode

The Factory Mode feature can be used to improve Program and Erse Times and it is managed through a diary location mechanism. If location UTEST at offset +0x0020 is found virgin at the moment of the reset, MCR.FERS can be set, after rising MCR.ERS and before MCR.EHV. After setting MCR.EHV the FPEC recognizes the value of FERS and allows a fast Program and Erase Operation. Under those conditions a factory Mode can start and no Read-While-Write is allowed (all the flh_busy sideband signals are set high). If the diary location has been programmed by the user (having changed from being virgin at reset), the Factory Mode gets permanently disabled.

35.4.6 Alternate program and erase interface

A second Modify interface is provided on the Flash for program and erase operations. The intent is that a dedicated master can have private access to blocks through this interface.

The alternate interface includes an alternate MCR register and alternate lock registers. Any block in low, mid, or high address space may be accessible through the alternate interface. The UTest NVM Block is also accessible for program through the alternate interface. Blocks in 256K Address Space are not accessible through this interface.

The alternate interface supports programming of data sizes from one doubleword (64 bits) to four doublewords within a single page (256 bits). Within a page, up to eight words may be altered in a single program operation. It is recommended that programming operations through the alternate interface be from 64 bits to 256 bits, and be 64-bit aligned. Programming has the same constraints as for the Main interface concerning ECC boundaries: 64b for Data Flash and 128b for Code Flash.

The alternate interface supports erasing a single block at a time with each erase operation.

As with the main interface, erase suspend, program suspend, and erase-suspended programs are supported.

Program and Erase times for the alternate interface are the same as the main interface when programs and erases are done in isolation. If main interface and alternate interface are being utilized simultaneously, bandwidth sharing is utilized.

Note: It is possible for the UTest NVM block to exist in both the main program interface and alternate program interface. In the event of a conflict the first interface to do the program to an address will get priority, and all other attempts after this will result in an OTP over-program failure (MCR[PEG] or MCRA[PEGa] being set to 0).

35.4.6.1 Alternate program interface

The programming procedure follows exactly the same flow as the main interface, except that only page programming is allowed.

The alternate interface program is done at lower priority compared to the main interface. Based on the size of programming allowed, the main interface has a 4:1 ratio advantage over the alternate interface. Program operations are allowed to finish on the active interface prior to allowing the inactive interface to execute its operation. Suspends done on the active interface enable the inactive interface to begin its operation.

For alternate program while erase, there could be a latency of up to 5 ms in the program operation.

For more information on interaction between the alternate program interface and the main interface, see *Section 35.4.6.3: Alternate interface performance*.

If the alternate interface is the only interface active, then normal program times may be expected.

35.4.6.2 Alternate erase interface

For erases done through the alternate interface, the single block to be erased is selected based on the address that is received during the erase interlock write. Thus the Select registers are not valid for the alternate interface (step 2 of the erase flow in *Section 35.4.5.2: Erase*), and the interlock write address is used (step 3 of the erase flow in *Section 35.4.5.2: Erase*). All other features of the main interface apply to the alternate erase interface. The erase procedure is exactly the same as for the main interface. See *Section 35.4.5.2: Erase* for more information on erase.

The alternate interface erase is done at lower priority compared to the main interface. The time-slice ratio between the two interfaces is 10:1, with 50 ms of time given to the main erase for every 5 ms of alternate interface. Suspends done on the active interface enable the inactive interface to begin its operation.

For alternate erase while program, there can be a latency of up to 5 ms in the main program time. Once the alternate erase is interrupted, it will remain interrupted for main programs for a fixed time of 50 ms, at which time the alternate erase will restart. It should be noted that multiple interruptions of the alternate erase will significantly increase the total erase time.

For alternate erase while erase, the main erase has priority and receives 50 ms of erase for every 5 ms of alternate erase. This may result in a significant increase in the erase time for the alternate interface, and a minor increase in erase time for the main interface.

For more information on interaction between the alternate erase interface and the main interface, see *Section 35.4.6.3: Alternate interface performance*.

If the alternate interface is the only interface active, then normal erase times may be expected.

35.4.6.3 Alternate interface performance

Table 456 highlights the interactions between the alternate interface and the main interface for program, erase, suspend, and idle conditions.

Table 456. Alternate Program and Erase Characteristics

Alt PGM/ERS interface request	Main PGM/ERS interface request	Alt interface result on memory array	Main interface result on memory array
Idle	Idle	Idle	Idle
Idle/SUSP	PGM/ERS	Idle/SUSP	PGM/ERS
PGM/ERS	Idle/SUSP	PGM/ERS	Idle/SUSP

Table 456. Alternate Program and Erase Characteristics (continued)

Alt PGM/ERS interface request	Main PGM/ERS interface request	Alt interface result on memory array	Main interface result on memory array
PGM	PGM	PGM	Priority PGM (4:1 ratio)
PGM	ERS	PGM with 5ms latency	ERS (possible minimal increase to total ERS time)
ERS	PGM	ERS (significant increase in total ERS time expected)	Priority PGM (possible latency of 5 ms) Once PGM is granted, 50 ms time slice available for more programs
ERS	ERS	ERS (significant increase in total ERS time expected)	Priority ERS (10:1 ratio) (possible minimal increase to total ERS time) 50 ms Main ERS for every 5 ms alternate ERS

35.4.7 User Test mode

Customers can put the Flash Memory module in User Test mode to perform specific tests for the integrity of the Flash array.

Three kinds of test can be performed:

- Array Integrity Self Check
- Margin Read
- Vth Distribution

User Test mode is an exclusive operation—it cannot be run in conjunction with any other Flash mode (Read, Write, Low-Power and Power-Down), and Read-While-Write functionality is inapplicable. Read accesses attempted by the user during User Test mode generates a Read-While-Write Error (RWE of MCR set). Vth Distribution mode is conditioned by SoC Life Status (mode allowed only as Failure analysis)

It is not permitted to perform User Test operations on the Test blocks (including the UTEST section).

35.4.7.1 Array Integrity Self Check

Array integrity is checked using a predefined address sequence (proprietary), and this operation is executed on selected (SEL0, SEL1, SEL2 and SEL3) blocks. Blocks marked by TMDBS are automatically excluded from MISR signature computation. Any random or non-random code is valid. Once the operation is completed, the results of the reads can be checked by reading the MISR value (stored in UM0–UM9), to determine if an incorrect Read, or ECC detection was noted. Array integrity requires that the Read Wait States and Address Pipelined control registers in the BIU be set to match the frequency being used.

The internal MISR calculator is a 64 + 8 + 1-bit register.

The 256 data, the respective ECC parity data, and the uncorrectable ECC (ERR) errors of the four doublewords are therefore captured by the MISR through 4 different Read accesses at the same location.

The whole check is done through 4 complete scans of the memory address space:

1. The first pass scans only bits 63-0 + 8 ecc + ERR of each doubleword.
2. The second pass scans only bits 128-64 + 8 ecc + ERR of each doubleword.
3. The third pass scans only bits 191-129 + 8 ecc + ERR of each doubleword.
4. The fourth pass scans only bits 255-192 + 8 ecc + ERR of each doubleword.

The 256-bit data and the respective ECC parity data are sampled after any single-bit/double-bit ECC correction and/or the uncorrectable error flag is set after any ECC detection. Single-bit and Double-bit corrections are always active and logging may be enabled (by UT0.SBCE1 and UT0.SBCE), resulting in the MCR.SBC and MCR.SBC1 flags and related address to be captured in ADR register, but this does not change the final signature in the UM0–UM9.

Only data from existing locations are captured by the MISR. The MISR can be seeded to any value by writing the UM0–UM9 registers.

The Array Integrity Self Check consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1’s to the appropriate field(s) in SEL0, SEL1, SEL2 and SEL3 registers. Blocks selected for Array Integrity Check do not need to be unlocked. It is not possible to perform Array Integrity operations on the UTEST Nonvolatile Memory block.
3. If desired, UT0.AIS can be set for sequential addressing only.
4. Seed the MISR UM0–UM9 with desired values.
5. If breakpoints are desired, set UT0.AIBPE to ‘1’, and ensure that MCR.EER, MCR.SBC and MCR.SBC1 are cleared. If breaking on single-bit correction is required, ensure that UT0.SBCE1 is set.
6. Write a logic ‘1’ to UT0.AIE to start the Array Integrity Check. Array integrity operations may be aborted prior to UT0.AID going high. This may be done by clearing UT0.AIE. It should be noted that in the event of an aborted Array Integrity check, the MISR registers contain a signature for the portion of the operation that was completed prior to the Abort, and is not definitive. Prior to performing further Array Integrity operations, the UM0–UM9 registers may require initialization to the desired seed value by performing register writes. Array Integrity operations may be suspended while UT0.AID is low. UT0.AISUS may be set to request an Array Integrity Suspend. Once UT0.AID has been set high, UT0.AISUS may be cleared to resume the Array Integrity sequence.
7. Wait until UT0.AID goes high.
8. Check that UT0.NAIBP is set to ‘1’ if breakpoints were previously enabled so that the MCR.EER, MCR.SBC, MCR.SBC1 and ADR registers may be checked to determine the cause and address of the break. Prior to resuming the operation, it necessary to clear MCR.SBC1, MCR.SBC or MCR.EER in order to preserve coherency of the flags and ADR updates with respect to any subsequent breakpoints. Operation may then be resumed by clearing UT0.NAIBP. Continue waiting until the UT0.AID bit goes high. If breakpoints were not enabled, or if UT0.NAIBP is low when UT0.AID goes high, then the operation is complete.
9. Compare UM0–UM9 content with the expected result.
10. Write a logic ‘0’ to the UT0.AIE bit.

Use sequential addressing for normal integrity checks of the Flash memory. If a more detailed check of the Read path is required (in diagnostic mode), leave UT0.AIS at ‘0’ and use the proprietary address sequence that checks the Read path more completely, however this sequence takes more time.

Caution: Do not modify the content of Block Select (SEL0, SEL1, SEL2 and SEL3) and Lock (LOCK0, LOCK1, LOCK2 and LOCK3) registers during execution of the Array Integrity Check operation or the MISR value may vary unpredictably.

User mode array reads requested during the Array Integrity test are ignored to ensure that the Array Integrity operation is not corrupted. The memory array does not respond to array Read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

While UT0.AID is low and UT0.AIE is high, the user may clear AIE, resulting in an Array Integrity Check Abort. UT0.AID must be checked to know when the aborting command has completed.

Array Integrity Check can be suspended by setting UT0.AISUS while Array Integrity Check is running (UT0.AID=0). Once suspended (UT0.AID=1) Array Integrity Check cannot be aborted and clearing UT0.AISUS starts the Resume.

Figure 365. Array Integrity Check of blocks (example)

```
UM0/9 = 0x0000_0000; /* Reset UM0/9 content */
UT0 = 0xF9F9_9999;   /* Set UTE in UT0: Enable User Test */
SEL0 = 0x0006_0000;  /* Set the corresponding bit in SEL0: Select Blocks
*/
SEL2 = 0x0000_0006;  /* Set the corresponding bit in SEL2: Select Blocks
*/
UT0 = 0x8000_0002;   /* Set AIE in UT0: Operation Start */
do                   /* Loop to wait for AID=1 */
{ tmp = UT0;         /* Read UT0 */
} while ( !(tmp & 0x0000_0001) );
```

35.4.7.2 Margin Read

User Margin Read may be achieved using the Array Integrity interface and has all the associated features of the Array Integrity interface (MISR calculation, suspend capability and Breakpoints).

Margin Read procedure (either Margin 0 or Margin 1) can be run on selected blocks in order to unbalance the Sense Amplifiers with respect to standard Read conditions, so that all the Read accesses reduce the margin vs ‘0’ (UT0.MRV=0) or vs ‘1’ (UT0.MRV=1).

User Margin Read is a self timed event and is independent of system clocks or wait states selected. Margin Read can be run on selected (SEL0, SEL1, SEL2 and SEL3) blocks, generating checksum calculations on MISR and ECC flagging. Blocks marked by TMDBS are automatically excluded from MISR signature computation. Once the operations are completed, the results of the Margin Reads can be checked, comparing checksum values in UM0–UM9 and reading MCR.EER and MCR.SBC to determine if zero, one or two bits are being detected by the Margin Read.

The MISR can be seeded to any value by writing the UM0–UM9 registers.

As with Array Integrity Check, the internal MISR calculator is a 64 + 8 + 1-bit register. The 256-bit data, the corresponding ECC parity data and the uncorrectable ECC (ERR) errors of the four doublewords are therefore captured by the MISR through 4 different Read accesses at the same location. The whole check is accomplished through 4 complete scans of the memory address space:

1. The first pass scans only bits 63-0 + 8 ecc + ERR of each doubleword.
2. The second pass scans only bits 128-64 + 8 ecc + ERR of each doubleword.
3. The third pass scans only bits 191-129 + 8 ecc + ERR of each doubleword.
4. The fourth pass scans only bits 255-192 + 8 ecc + ERR of each doubleword.

The 256-bit data and the respective ECC parity data are sampled after any single-bit ECC correction, and the double error flags are set after any ECC detection. Single-bit and Double-bit corrections are always active and logging may be enabled (by UT0.SBCE1 and UT0.SBCE), resulting in the MCR.SBC and MCR.SBC1 flags and related address to be captured in ADR register, but this does not change the final signature in the UM0–UM9.

In any case, the charge losses detected through the Margin Read cannot be considered device failures and no Failure Analysis is initiated on them.

The Margin Read Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers. Blocks selected for Margin Read do not need to be unlocked. It is not possible to do Margin Read operations on the UTEST Nonvolatile Memory block.
3. Seed the MISR UM0–UM9 with desired values.
4. Ensure that MCR.EER, MCR.SBC and MCR.SBC1 are cleared.
5. Set UT0.SBCE1 to '1' to detect single bits during Margin Read. Set UT0.SBCE to '1' to detect double bits during Margin Read.
6. Change the value of UT0.MRE from '0' to '1'.
7. Select the Margin level: UT0.MRV=0 for 0's margin, UT0.MRV=1 for 1's margin.
8. Write a logic '1' to UT0.AIE to start the Margin Read.
9. Wait until UT0.AID goes high.
10. Check that UT0.NAIBP is set to '1' if breakpoints were previously enabled so that MCR.EER, MCR.SBC and ADR.ADDR may be checked to determine the cause and address of the break. Prior to resuming operation, clear MCR.SBC1, MCR.SBC or MCR.EER. Operation may then be resumed by clearing UT0.NAIBP. Continue to wait until UT0.AID goes high. If breakpoints were not enabled, or if UT0.NAIBP is low when UT0.AID goes high, then the operation is complete.
11. Compare UM0–UM9 content with the expected results.
12. Check MCR.EER, MCR.SBC and MCR.SBC1.
13. Write a logic '0' to UT0.AIE, UT0.MRE and UT0.MRV.
14. If more blocks are to be checked, return to step 2.

The linear address sequence is used regardless of the value of UT0.AIS.

Caution: Do not modify the content of Block Select (SEL0, SEL1, SEL2 and SEL3) and Lock (LOCK0, LOCK1, LOCK2 and LOCK3) registers during execution of the Margin Read operation, otherwise the MISR value may vary unpredictably.

User mode array reads requested during the Margin Read are ignored to ensure that the Margin Read operation is not corrupted. The memory array does not respond to array Read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

While UT0.AID is low and UT0.AIE is high, the user may abort Margin Read by clearing AIE. UT0.AID must be checked to know when the aborting command has completed. Margin Read can be suspended by setting UT0.AISUS while Margin Read is running (UT0.AID=0). Once suspended (UT0.AID=1), Margin Read cannot be aborted and clearing UT0.AISUS starts the Resume.

Figure 366. Margin Read Check versus 1's

```
UM0/9 = 0x0000_0000; /* Reset UM0/9 content */
UT0 = 0xF9F9_9999;   /* Set UTE in UT0: Enable User Test */
SEL0 = 0x0006_0000;  /* Set LSL2-1 in SEL0: Select Blocks */
UT0 = 0x8000_0020;   /* Set MRE in UT0: Select Operation */
UT0 = 0x8000_0030;   /* Set MRV: Select Margin versus 1's */
UT0 = 0x8000_0032;   /* Set AIE in UT0: Operation Start */
do                   /* Loop to wait for AID=1 */
{ tmp = UT0;         /* Read UT0 */
} while ( !(tmp & 0x0000_0001) );
data0/9 = UM0/9;     /* Read UM0/9 content*/
UT0 = 0x8000_0030;   /* Reset AIE in UT0: Operation End */
UT0 = 0x0000_0000;   /* Reset UTE, MRE, MRV, AIS in UT0:
                        Deselect Operation */
```

35.4.7.3 Vth Distribution

Vth Distribution functionality can be used in case of Failure Analysis for quick assessment of cells status in terms of Vth. The functionality has bit register UTO.VTD as initiator. Writing of this bit is enabled by the Life Cycle state of the device. Life Cycles has to be in Failure Analysis (FA) condition to allow Vth Distribution mode.

Vth Distribution functionality works on selected blocks on the base of multi internal runs. Each run corresponds to an internal voltage step, where cells read as '1' (Vth lower than internal voltage level) are counted: the completion of each run provides on UMISR0 the count of these cells. Clearing UTO.NAIBP starts next voltage step run, till completion of the intended voltage range. Number of internal voltage step is fixed. Aim of the test is to sweep the range from the erased cells with the highest Vth to the programmed cells with the lowest Vth, therefore to provide an assessment of the effective distribution of the cells in the selected blocks.
Once all the cell counts, one per run, have been collected, distribution can be obtained as difference between counts collected through subsequent steps.
In every run, all the cell, including the ECC parity ones, are counted for the block selected.
Vth Distribution results are expected to be collected at room temperature (25C).

Vth Distribution Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers.
3. Unlock blocks Blocks o be checked by writing 0's to the appropriate register(s) in LOCK0, LOCK1, LOCK2 and LOCK3 registers. Blocks selected for Vth Distribution need to be unlocked. It is not possible to do Vth Distribution operations on the UTEST Nonvolatile Memory block, neither on blocks protected through TMD SEAL Block Select.
4. Set UT0.VTD to '1' to select Vth Distribution mode.
5. Write a logic '1' to UT0.AIE to start the first internal run.
6. Wait until UT0.AID goes high.
7. Check that UT0.NAIBP is set to '1'. If NAIBP is at '1' it means current internal run has been completed and device is ready to increase voltage step. If NAIBP is at '0' the overall sweep has been completed.
8. Collect UM0 content for the current voltage step (run).
9. Clear UT0.NAIBP to increase voltage step and return to 6)

Caution: Voltage levels used during Vth distribution sweep may alter the intended reliability of the Cells as data retention. For this reason Vth distribution can be used no more than 10 times.

35.4.8 Protection strategy

Two kinds of protection are available: Modify Protection to avoid unwanted program/erase in Flash blocks and Test Mode Disable to avoid piracy.

35.4.8.1 Modify Protection

The Flash Modify Protection information is stored in nonvolatile Flash cells located in the UTEST region. This information is read once during the Flash Initialization phase following exit from reset and is stored in volatile registers that act as actuators at SoC level. The information is passed to the Flash Memory module through erase_lock and program_lock sidebands, and allows independent protection of each block of Low, Mid, High and 256K Address Space Block.

There are also Address Space Block Locking registers provided to independently lock/unlock each block of Low, Mid, High and 256K Address Space Block against program and erase. Locking is done through the LOCK0, LOCK1, LOCK2 and LOCK3 registers. These registers do not have a nonvolatile image stored in the UTEST area of TestFlash, so the locking information is kept on reset: volatile content is all 1's, meaning all blocks are locked.

All the volatile registers can be written to '0' or '1' at any time, therefore the user application can lock and unlock blocks when desired. Note, however, that overall protection is achieved by the combination of sidebands with Low, Mid, High and 256K Address Space Block Locking registers in such a way that a block can be programmed/erased only if both (sidebands and Locking registers) indicate an unlocked state. Sidebands independently control program protection, erase protection and over-programming protection (OPP) on a block-by-block basis.

35.4.8.2 Test Mode Disable

In the TestFlash area a mechanism is also available to disable entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected for protection in this way cannot be analyzed for possible failures by manufacture failure analysts.

Protection of this sort prevents all high voltage operations to the Flash executed by the internal FPEC as well as reads from FPEC, including Array Integrity Check or Margin Read, when using Test mode interfaces. Protection through other interfaces is managed through normal User mode protection mechanisms, that are System specific.

To disable manufacturer entry into Test mode, first program the Test Mode Disable Seal location to 0x5A4B_3C2D and once the next reset is asserted, Test mode is disabled.

It is possible to create a password to enable manufacturer entry into Test mode. This can be programmed into the Test Mode Disable Override Passcode. It is safest to do this prior to censoring the part to avoid unintended lockout. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and are not accepted for override. If the customer desires that override never be possible, one of the three invalid passcodes should be put into this location. Passcodes may be entered to authenticate entry (if enabled) by performing a register Write to offset address 0x90.

Only blocks selected in the Test Mode Disable Block Select field are controlled by the Test Mode Disable feature. Thus is it possible for customers to select blocks for this type of protection, and render them ineligible for manufacturer failure analysis. Bits programmed to '0' in the Test Mode Disable Block Select field designate blocks that are controlled by Test Mode Disable feature. The TestFlash block, and thus the UTEST and BAF sections, is always protected once the Test Mode Disable Seal password is written. In order to permit two opportunities to select blocks for Test Mode disable, two regions are available, both in UTEST: one at offset 0x50, and the other at offset 0x60; both are 128-bit sized (only some may be effective, depending on actual sectorization). The bits of these two regions are logically ANDed in order to define which blocks are effectively selected for the Test Mode Disable feature.

The bit Block Select field is organized as follows, and aligned in the way blocks are defined in the LOCKx/SELx registers:

Table 457. Test Mode Disable block select

Block	Block selection bits
Blocks in Mid space	Data[16:31]
Blocks in Low space	Data[2:15]
Blocks in High space	Data[48:63]
Blocks in 256K space	Data[64:127]

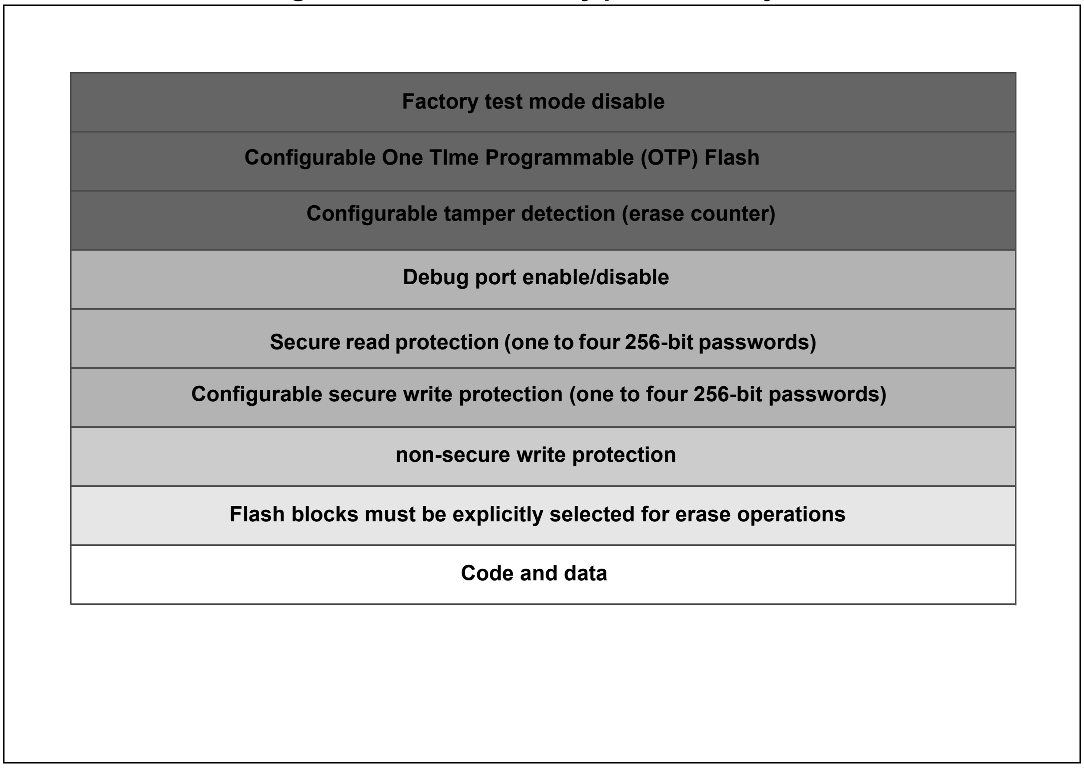
36 Flash Memory Programming and Configuration

This chapter provides basic guidance on activities related to configuring and programming the embedded Flash memory in the SPC584Bx microcontroller.

Multiple layers of protection are available to prevent unintended updates to Flash contents and unauthorized reads and writes to Flash contents. The protection mechanisms range from simple write-protect bits for individual Flash blocks to four levels of 256-bit password read/write protection and configuration of Flash memory blocks as One Time Programmable (OTP).

Additionally, the SPC584Bx microcontroller includes a tamper detection feature, debug port enable/disable, and the ability to disable the ability for the device to be put into factory test mode.

Figure 367. Flash memory protection layers



At the lowest layer of protection, all Flash erase operations require the explicit selection of one or more blocks of Flash memory before the operation is performed. When executed, the operation is only performed on the selected block(s).

The next layer is the chip's ability to lock individual Flash blocks against unintended programming and erase operations. This level of protection does not offer any password protection and is intended to provide protection against *accidental* changes to code and data.

The first layer of *secure* protection is the microcontroller's ability to support up to four levels of 256-bit password-secured *write* protection for individual Flash blocks.

Similarly, up to four levels of password-secured *read* protection can be implemented. Unlike the password-secured write protection, which is implemented on a per-block basis, password-secured read protection applies to five pre-defined (at the factory) groups of Flash blocks. The read protection applied to a group applies to each block within the group. In contrast, secure write protection is defined at the individual block level.

Secure password protection can also be implemented for the debug port.

The SPC584Bx microcontroller also includes a tamper detection feature. In its most basic implementation, tamper detection acts as an erase counter. If tamper detection is implemented, a DCF record in the tamper detection *diary* must be written by software before a protected Flash block can be erased. Multiple tamper detection regions can be defined by the customer, with each region containing one or more blocks. A block can be included in more than one tamper detection region if desired.

Another protection feature related to tamper detection is the ability to configure any Flash block as one-time programmable (OTP).

Finally, the microcontroller includes a feature that allows the customer to disable the ability to place the device in factory test mode.

The remainder of the chapter provides details regarding these protection mechanisms and gives a brief overview of security configuration planning.

Note: Most of the topics in this chapter build on information presented in previous topics. To get a clear overview of how the Flash protection features work and relate to each other, please read the chapter in sequence from beginning to end.

Access protection configuration is applied on a per-block basis (except read protection and tamper detection applied to groups of Flash memory blocks), which means that a custom level of access protection can be provided for each Flash memory block. This is accomplished through the use of registers containing bits that are mapped to individual Flash blocks. Understanding the mapping is key to understanding the implementation and use of the SPC584Bx microcontroller's Flash protection features.

Depending on the type of protection to be applied, different sets of registers (or even DCF records) are used, but the mapping scheme is consistent. This mapping scheme is first presented in *Section 36.1: Selection of Flash memory blocks for erase*. It is important to understand the topic before proceeding to other sections.

36.1 Selection of Flash memory blocks for erase

The registers used for selecting Flash blocks for erase operations are the Flash control registers named SEL0, SEL1, SEL2 and SEL3.

The SEL0–2 registers are 32-bit registers containing fields associated with specific address spaces: low, mid, high, and 256 KB. Each non-reserved bit in the fields is mapped to a specific Flash block.

For example, *Section 35.3.2.10: Select 0 register (SEL0)* shows the Select 0 (SEL0) register for the Flash module. This register provides a way of selecting blocks in the low and mid address spaces for erase operations.

Table 402: Flash memory partition 0 memory map shows an example of a mapping of SEL0 register's LOWSEL and MIDSEL field bits to individual Flash blocks. In this case, not all bits of the LOWSEL and MIDSEL fields are mapped—only LOWSEL bits 0–11 (register bits 15–4) and MIDSEL bits 0–2 are mapped. Before any mapped block can be erased, it must be

selected by writing a ‘1’ to its corresponding bit in one of the SEL*n* registers. Refer to the device configuration chapter for the complete mapping.

Note: Values written to the SEL registers do not persist through device reset. As indicated by the reset values shown in Figure 327: Select 0 register (SEL0), all bits are reset to ‘0’. As a result, after a device reset all blocks mapped to the register are, by default, not selected for erase.

Example 1 Selecting blocks for erase

Writing the value 0x0005_0000 to the SEL0 register (*Figure 327: Select 0 register (SEL0)*) selects the blocks in *Table 417: SEL0 field descriptions* for erase.

36.2 Non-secure write protection

Non-secure write protection refers to the chip’s ability to lock individual Flash blocks against programming and erase operations. This level of protection is intended to provide protection against *accidental* changes to code and data. Although similar in function, this is *not* associated with the built-in register protection (REG_PROT) feature included with many of the chip’s modules. The non-secure write protection feature detailed here applies only to embedded Flash memory.

Implementing non-secure write protection is almost identical to selecting specific Flash blocks for an erase operation—the only difference is the register used. To provide non-secure write protection for a block, write a ‘1’ to the corresponding mapped bit found in one of the LOCK0–3 Flash control registers.

Note: As indicated by the reset values shown in Section 35.3.2.4: Lock 0 register (LOCK0), after reset all blocks mapped to the register are, by default locked against programming and erase. Values written to the LOCK and SEL registers do not persist through device reset.

The same bit mapping that applies to the SEL0–3 registers also applies to the LOCK0–3 registers.

Example 2 Selecting blocks for write protection

Writing the value 0x0005_0000 (same value used in *Example 1*) to the LOCK0 register (*Section 35.3.2.4: Lock 0 register (LOCK0)*) locks the blocks in *Table 411* (same blocks selected in *Example 1*) so they cannot be programmed or erased, even if selected for erase.

As long as the LOCK0[LOWLOCK] bits are unchanged, the above blocks cannot be modified unless the protection is overridden (discussed in *Section 36.3: Secure write protection*). All other blocks mapped to bits in the LOCK0 *can* be programmed because their corresponding bits have been set to ‘0’.

36.3 Secure write protection

Secure write protection refers to the chip’s ability to implement up to four levels of 256-bit password-secured *write* protection for individual Flash blocks. Use of this feature is optional but the default behavior of the SPC584Bx microcontroller is to apply secure write protection. If there is a need to *not* write-protect specific blocks, that needs to be configured by creating the appropriate DCF records. For details, please refer to *Section 78.4.2.1: JTAG Password Challenge* in the PASS chapter.

Note that the *secure* write protection settings override the non-secure write protection detailed in *Section 36.2: Non-secure write protection*.

Implementing a custom secure write protection scheme requires creating DCF records that define the level of write protection to be applied to each Flash block. The DCF records use a bit mapping scheme identical to the mapping in the Flash module's SEL*n* and LOCK*n* registers. The write protection scheme becomes a permanent part of the device but blocks can be temporarily unlocked. Depending on the specific implementation, temporarily unlocking the write protection requires matching up to four 256-bit passwords.

The easiest way to understand the secure write protection functionality is to examine the role played by the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers are four identical sets of registers contained in the PASS module—they distinguished by the PG*n* suffix. The PG*n* (Password Group *n*) suffix indicates the password required to unlock that set of registers. Password 0 unlocks PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. Similarly, the registers suffixed with PG1 are unlocked by matching password 1, and so on.

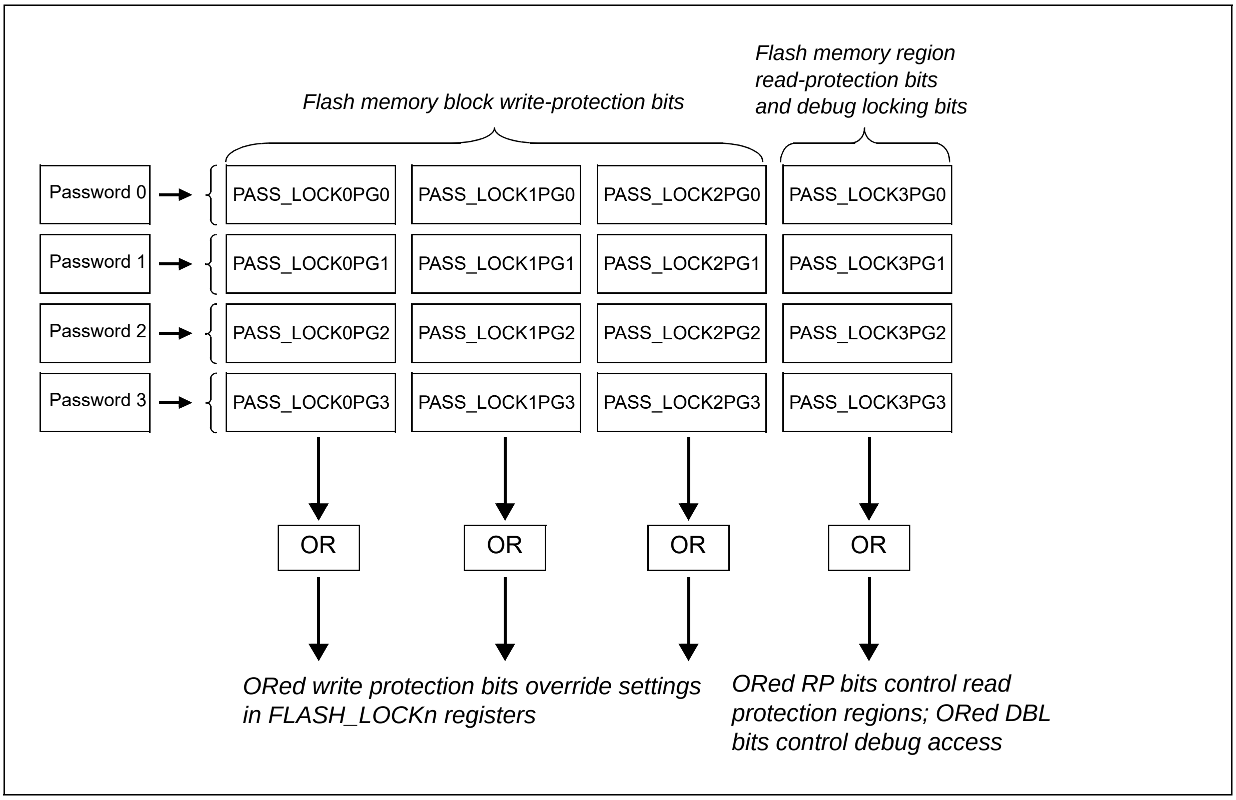
The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* register sets have two functions related to secure write protection:

- Reading the registers provides the secure write lock status for each Flash block.
- Changing the value of a mapped bit in a register changes the secure write protect status of the corresponding Flash block.

Each set of registers contains the same mapping of bits to Flash blocks as the Flash module's SEL*n* and LOCK*n* registers. In any of the four PASS_LOCK*x*_PG*n* register sets, if a mapped bit is set to '1', locking a Flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access to the register, and setting the bit to '0'. If the same bit is set to '1' in more than one PASS_LOCK*x*_PG*n* register set, each must be unlocked using the corresponding password and the bit must be set to '0' before the Flash block is available for programming or erase.

Figure 368 shows how the registers interact. Note that secure write protection is only affected by the settings in PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n*. The PASS_LOCK3_PG*n* registers have additional security functions, discussed in later topics.

Figure 368. Secure write protection



The next sections discuss implementing secure write protection and overriding secure write protection.

36.3.1 Implementing secure write protection

Implementing secure write protection is a one-time task performed either at the factory or by the customer during chip configuration activities. It normally requires creating a DCF record for each of the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers[k]. The DCF records have a structure that includes the same mapping of bits to Flash memory blocks as the Flash module's SEL*n* and LOCK*n* registers. The values assigned become the permanent register reset values for the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers.

Refer to *Chapter 9: Device Configuration Format (DCF) Records* for complete details on DCF records. This section summarizes the contents of a DCF record as required to implement secure write protection.

Even though the mapping of bits to Flash memory blocks is identical to the mapping used for selecting Flash blocks for erase of non-secure write locking, there are two significant differences:

1. Configuring Flash blocks for secure write protection is done only during device configuration to customer specifications, either at the factory or by the customer during

k. The default behavior for secure write protection in the case where passwords are created but no secure write protection DCF records are created, is for all blocks to have four levels of password-secured write protection. To vary the protection among all blocks, a DCF record for each password group must be created.

OEM production activities. The selection becomes a permanent part of the device configuration.

2. Configuration of blocks for secure write protection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

Caution: Careful planning is required before writing DCF records. They are created in a write-once area of Flash. Multiple instances of any DCF record can be created, but in some cases only the record created first is used and with others only the most-recently written record is used. Be sure to understand the characteristics of a DCF record before creating it.

The DCF records required for implementing a secure write protection scheme are detailed in the PASS section of DCF client list table (*Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*). Table entries provide the CS and Address value combinations that correspond to PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The format of a secure write protection record is summarized here for convenience.

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST Flash memory. The structure of a DCF Record is shown in *Figure 369*.

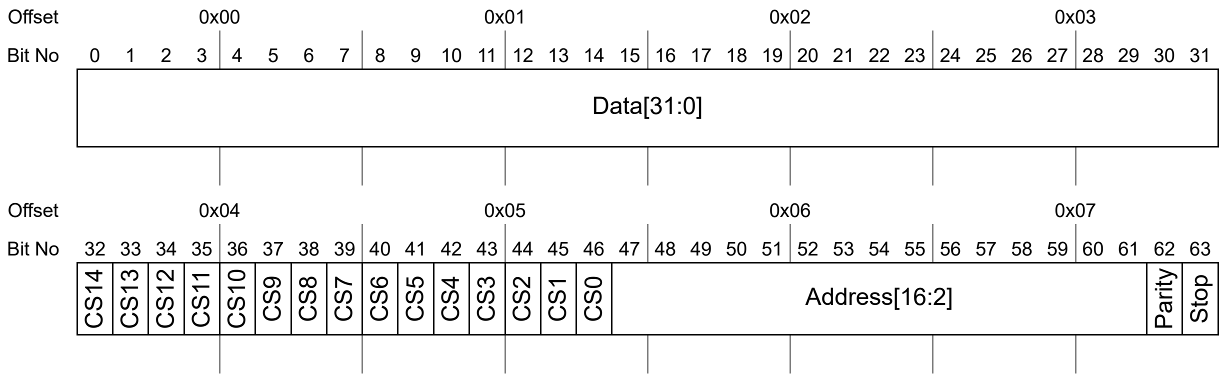


Figure 369. DCF record structure

Note: *The values in the PASS_LOCKxPGn registers should mirror the settings in the Flash module's LOCK0–3 registers.*

Values written to the PASS_LOCKxPGn registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated Flash block to be locked against programming or erase.

The PASS_LOCKxPGn bits take effect when the life cycle is matured to "OEM Production" or older.

Example 3 Implementing secure write protection

Implementation of secure write protection is done during initial Flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) must be stored in a protected OTP area of device Flash called UTEST Flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n*) that have the same mapping of register bits to Flash blocks as the Flash module's SEL0–3 registers and LOCK0–3 registers.
2. DCF records must be created to define which Flash memory blocks are to be locked and which blocks are not locked.

The records defined in step 2., determine the reset values of the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers and therefore the write protection settings that take effect each time the microcontroller is reset.

Assume four passwords have been previously implemented and the following blocks are to be locked in the password 0 group: refer to Table 411

From the PASS section of Table 107: DCF client list in Chapter 9: Device Configuration Format (DCF) Records:

Table 458. DCF client list of PASS

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0000	LOCK0_PG0

Referring to Figure 369, the values for the DCF record are:

- Data[31:0] is 0x0005_0000 (bits 0–31) (same value used in previous examples)
- CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
- Address[16:2] is 0b000_0000_0100_0000 (bits 47–61)
- Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
- Preferred Stop value is 0 (refer to footnotes in Figure 370).

The following DCF record provides secure write protection for the blocks by setting the reset values PASS_LOCK0_PG0 to the appropriate value to lock the blocks.

Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit No	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	X (1)	X (2)

Figure 370. Sample DCF record for secure write protection

1. Bit 62 indicates parity strategy, which is not used in the PASS DCF records so it is a “don’t care”.
2. Bit 63 indicates whether the current record is the last DCF. Generally, this should be written as ‘0’ so the first unprogrammed (all 1’s) DCF record acts as the stop record.

The remaining blocks mapped to the PASS_LOCK0_PG*n* registers do *not* have secure write protection provided by the password 0 register set because the DATA field of DCF record has only the bits mapped to Flash blocks 4 and 5 set to ‘1’.

Caution: The above example is intended to highlight the need to carefully understand how DCF records work. The protected blocks are secured by password 0 but by default they are also secured by passwords 1, 2, and 3. In contrast the blocks *not* secured by password 0 in this example *are* secured by passwords 1, 2, and 3.

Refer to the PASS section of DCF client list table (Table 107: DCF client list in Chapter 9: Device Configuration Format (DCF) Records) for details on the DCF records.

36.3.2 Overriding secure write protection

The “PG*n*” (Password Group *n*) suffix on each the register names indicates the password each set of registers is associated with. For example, password 0, is used to unlock

PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. The bits in the first three registers (PASS_LOCK0_PG0, PASS_LOCK1_PG0, and PASS_LOCK2_PG0) are mapped exactly the same as the bits in the Flash module's SEL0–2 registers and LOCK0–2 registers.

Like the SEL0 and LOCK0 registers, the PASS_LOCK0_PG*n* registers all contain bits mapped to the low and mid address spaces, and the mapping scheme is the same. A mapped bit set to '1' in any of the registers causes the corresponding block to be locked against program or erase. A mapped bit set to '0' in any of the registers makes the corresponding block available for programming or erase.

If a bit is set to '1', locking a Flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access, and setting the bit to '0'. If the same bit is set to '1' in more than one of the PASS_LOCK*x*PG*n* register sets, each must be unlocked using the corresponding password and the bit must be set to '0' before the Flash block is available for programming or erase.

Figure 371 shows the PASS_LOCK0_PG*n* register. It has the same structure as the Flash module's LOCK0 register. In addition to the main difference of being among a group of four parallel sets of locking registers, the other difference between the PASS_LOCK*x*_PG*n* registers and the Flash LOCK*n* registers is how the reset values are determined.

By default, all mapped bits are set to '1', locking their corresponding Flash blocks from programming and erase. The reset value is user-configurable, though, by writing a DCF record. This is discussed in a later section.

Offset: 0x0100 LOCK0_PG0, 0x0110 LOCK0_PG1, 0x0120 LOCK0_PG2, 0x0130 LOCK0_PG3 — Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TSLOCK	ATSL	LOWLOCK													
Reset	x(1)	0	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MIDLOCK															
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

Figure 371. LOCK0_PG*n* register

1. If no DCF records have been written to change the lock state, the bit will reset to '1'

Note: The values in the PASS_LOCKxPGn registers should mirror the settings in the Flash module's LOCK0–3 registers.

Values written to the PASS_LOCKxPGn registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated Flash block to be *locked against programming or erase. The purpose of the registers is to indicate secure write protection status and to provide a way to temporarily override secure write protection.*

The PASS_LOCKxPGn bits take effect after the life cycle is matured to “OEM Production” or older.

Example 4 Overriding secure write protection

Assume four passwords have been created but no DCF records have been written to set the reset values of the PASS_LOCK*x*_PG*n* registers, so the reset value for each mapped bit defaults to ‘1’, causing all mapped blocks to have four levels of 256-bit password-secured write protection.

Since the settings in the PASS_LOCK*x*_PG*n* registers override the settings in the Flash module’s LOCK0–3 registers, the settings in those registers are irrelevant. Each of the four sets of the PASS module’s LOCK registers must be unlocked by supplying the correct password for each set and then the appropriate bits must be set to ‘0’ to make the desired Flash blocks available for program or erase.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant 32 bits of the password written to PASS CIN0.
3. Write 0xFFCF_FFFF to the PASS_LOCK0_PG0 register and verify a successful write.
4. Repeat the above steps for passwords 1–3.

As in the previous examples, the result is the following blocks are available for program and erase operations but at a much higher level of security.

36.4 Secure read protection

Read Protection is ability of the Flash controller to block read accesses to the Flash memory from all bus masters. This includes the blocking of:

- Instruction fetches and load instructions from the CPU
- Debug reads from an outside tool
- DMA reads
- SIPI reads
- All other types of bus master reads

When secure read protection is active, the affected Flash blocks are readable by any bus master until a debugger is attached to the JTAG port and debug is enabled. If a debugger is attached and debug is enabled while an application is running, the next access to a Flash read protected region terminates with a bus error.

Most instances of Flash memory read protection in the SPC584Bx microcontroller are *not* configurable by the customer. An example of this is the 256-bit passwords stored in UTEST Flash memory. Password-based read protection *is* customer-configurable[l] and is similar to the password-based write protection mechanism detailed in *Section 36.3: Secure write protection*.

There are two significant differences between password-based read protection and password-based write protection in the SPC584Bx microcontroller.

1. Read protection can only be applied to pre-defined groups of Flash memory blocks. These groups, called “Read Lock groups”, cannot be re-defined, meaning the blocks assigned to those groups are permanently assigned and cannot be assigned to other groups and no new groups can be created.
2. Read protection is only active either when a debugger is attached to the JTAG port and debug is enabled; or the lifecycle has been advanced to “Failure Analysis”.

Another difference is that only the PASS_LOCK3_PG*n* register for each password group is used:

- PASS_LOCK3_PG0
- PASS_LOCK3_PG1
- PASS_LOCK3_PG2
- PASS_LOCK3_PG3

Note: The mappings of PASS_LOCK3_PGn register bits to Flash blocks vary with different microcontrollers. Refer to the Flash memory section in the Device Configuration chapter of this reference manual for the mappings specific to the SPC584Bx microcontroller.

The next sections discuss implementing secure read protection and overriding secure write protection.

36.4.1 Implementing secure read protection

Implementation of secure read protection is done during initial Flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) are stored in a protected OTP area of device Flash named UTEST Flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n*) that have the same mapping of register bits to Flash blocks as the Flash module’s SEL0–3 registers and LOCK0–3 registers. These are the same passwords used by secure write protection and other secure functions. The passwords are only created once.
2. DCF records must be created to define which Flash memory blocks are to be locked. Those records determine the reset values of the PASS_LOCK3_PG*n* registers.

   Like the other PASS_LOCK*x*_PG*n* registers, the PASS_LOCK3_PG*n* registers have two functions.
   - Reading the registers provides a lock status for each Read Locking group.
   - Changing the value of a mapped bit in a register changes the read protect status of the corresponding Read Locking group.

The layout of the PASS_LOCK3_PG*n* registers is shown in *Figure 372*. The fields relevant to implementing secure read protection are RL4–RL0.

l. Secure read protection is user-configurable in that the default lock/unlock state for each mapped block at reset is controlled by user-created DCF records, giving the user the ability to determine the number of levels of password protection implemented. The read protection is applied to groups of blocks instead of individual blocks and the groups are not user-definable.

Offset	0x010C LOCK3_PG0 0x011C LOCK3_PG1 0x012C LOCK3_PG2 0x013C LOCK3_PG3
Access: Read/Write	

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	RL4	RL3	RL2	RL1	RL0
W																
Reset	x(1)	x(2)	x(2)	0	1	1	1	1	0	0	0	x(2)	x(2)	x(2)	x(2)	x(2)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	256LCK_U															
Reset	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)

Figure 372. PASS_LOCK3_PG*n* Register

1. Reset value depends on life cycle. If life cycle is Customer delivery or earlier, the value will be 0, otherwise 1.
2. If no DCF records have been written to change the lock state, the bit will reset to ‘1’.

Example 5 Implementing secure read protection

Assume four passwords have been previously implemented.

Following is an example of a mapping of Flash blocks to Read Locking groups.

To provide two levels of password protection for only Read Lock group 3 and leave all other groups unlocked, the RL3 bit can be set in any two of the four PASS_LOCK3_PG*n* registers. For this example, we will choose password groups 0 and 1.

We need to create DCF records that define the reset values for all RL*n* bits as 0, with the exception of PASS_LOCK3_PG0 and PASS_LOCK3_PG1. The information required is from the PASS section of *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

Table 459. DCF client list of PASS

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0011	LOCK3_PG0
000_0000_0000_1000	000_0000_0100_0111	LOCK3_PG1
000_0000_0000_1000	000_0000_0100_1011	LOCK3_PG2
000_0000_0000_1000	000_0000_0100_1111	LOCK3_PG3

Referring to *Figure 369*, the values for the DCF record are:

- For LOCK3_PG0
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG1
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0111 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG2
  - Data[31:0] is 0x0000_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_1011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG3
  - Data[31:0] is 0x0000_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_1111 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.

36.4.2 Overriding secure read protection

As noted previously, the PASS_LOCK3_PG*n* registers (refer to *Figure 372*) contain five read protection fields, RP0–RP4, that correspond to pre-defined Read Lock groups. To temporarily unlock a Read Lock group, the appropriate PASS_LOCK3_PG*n*[RL*n*] bits must be set to 0.

Like the register bits controlling secure write protection, the reset value of bits controlling secure read protection is determined by DCF record or, in the absence of a relevant DCF record, each bit resets to ‘1’, causing the blocks associated with each read group to be read locked.

Example 6 Overriding secure read protection

Building on *Example 5: Implementing secure read protection*, where Read Lock group 3 was protected by passwords 0 and 1, temporarily overriding secure read protection for that group is accomplished by first unlocking the PASS_LOCK3_PG0 and PASS_LOCK3_PG1 registers and then setting the RL3 bit to ‘0’ in both.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write a 0 to PASS_LOCK3_PG0[RL3] bit and verify a successful write.
4. Repeat the above steps for password 1.

The result of a successful sequence is that the blocks associated with the unlocked read lock group can be read by a debugger.

36.5 Debug port enable/disable

The mechanism for temporarily enabling the SPC584Bx debug port after it has been locked by its life cycle state is similar to the Flash memory read protection mechanism detailed in *Section 36.4: Secure read protection*. The difference is that the PASS_LOCK3_PG*n*[DBL] field is used instead of the RP0–RP4 fields.

36.6 Tamper detection

The Tamper Detection Module (TDM) provides a type of Flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

This section provides details on the mapping between DCF record bits and individual Flash memory blocks to be selected for tamper detection.

Refer to *Chapter 79: Tamper Detection Module (TDM)*, for details on tamper detection, including the DCF clients and two additional memory-mapped registers.

There are two significant differences between selecting blocks for tamper detection and selecting blocks for other activities, such as a Flash erase operation:

1. Selection of blocks for tamper detection protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. The selection of blocks for tamper detection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

The Tamper Detect section of DCF client list table (*Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*) gives the CS and Address value combinations that correspond to groups of LOCK0–LOCK3 registers, similar to the Flash module's LOCK0–LOCK3 registers.

The mapping of individual Flash blocks to TDM DCF record data field bits is shown included in the detailed Flash block map in the Flash section of the Device Configuration chapter.

36.6.1 Implementing tamper detection

Implementing tamper detection requires:

1. Creating the tamper detect diary
2. Assigning blocks to Tamper Detection Regions (TDRs)

The tasks are detailed in the following sections.

36.6.2 Creating the tamper detect diary

Tamper detection in the SPC584Bx microcontroller is user-defined. The basic functionality provided is an erase counter, or diary, for 6 user-defined regions, called “tamper detect regions”, or TDRs, within the Flash memory. Each TDR is defined using a set of DCF records that is similar in function and layout to the Flash module’s SEL*n* and LOCK*n* registers, which contain bits matched to specific Flash blocks.

For a program or erase operation to be performed on a block included in a TDR, a record must be written to a “diary” that tracks Flash modification operations. The format and size of records is customer-defined. The only hardware constraints are that each diary has a maximum size of 4 KB and the minimum size of any programming operation depends on the Flash sector, 8 bytes for Data sector and 16 bytes for the remaining sector.

Note: The limiting factor for tamper detection records is the diary size. The total diary size is given by the number of TDRs (6 for SPC584Bx) multiplied by 4 KB, the portion of the diary allotted to records associated with each TDR.

The diary can be placed within any Flash block in the Flash array but to maintain the security of the TDR, there are two constraints:

- The block where the diary is placed must be assigned as OTP within the OTP registers (covered in the next section).
- The diary base address must be on a 4 KB boundary, therefore the least significant 12 bits of the base address must all be ‘0’.

Caution: The tamper detect diary base address DCF record is a “write once” record. All subsequent tamper detect diary base address DCF records created are ignored.

Example 7 Creating the tamper detect diary

If the total space required for a diary is 24 KB (as in the case of 6 TDRs), any Flash block in the SPC584Bx microcontroller but B0F0 and B0F1 whose size is not enough to allocate the diary total space can be used. Since the diary base address must be on a 4 KB, boundary, though, if the diary base address is within a 32 KB Flash block, it can be assigned to the base address of the block or the address given by the base address plus 4 KB or the address given by the base address plus 8 KB.

If the base address of the 32 KB block is 0x00FD000, the base address of the diary is given by:

32’b0000_0000_1111_1101_0000_0000_0000_0000 (32 KB base address)

or

32’b0000_0000_1111_1101_0001_0000_0000_0000 (32 KB base address + 4 KB)

or

32’b0000_0000_1111_1101_0010_0000_0000_0000 (32 KB base address + 8 KB)

The base address of each TDR's section of the diary is:

- TDR0 = Diary_Base_Address + 16'b0000_0000_0000_0000
- TDR1 = Diary_Base_Address + 16'b0001_0000_0000_0000
- TDR2 = Diary_Base_Address + 16'b0010_0000_0000_0000
- TDR3 = Diary_Base_Address + 16'b0011_0000_0000_0000
- TDR4 = Diary_Base_Address + 16'b0100_0000_0000_0000
- TDR5 = Diary_Base_Address + 16'b0101_0000_0000_0000
- End of the diary = Diary_Base_Address + 16'b0110_0000_0000_0000

We need to create one DCF record that defines the base address for the diary. The information required is from the Tamper Detect section of *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

Table 460. DCF client list of Tamper Detect

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0000	Diary Base Address

Referring to *Figure 369*, the values for the DCF record are:

- For LOCK0_PG0
  - Data[31:0] is 0x00FC4000 (the base address).
  - CS[14:0] is 0b000_0000_0001_0000 (bits 32–46).
  - Address[16:2] is 0b000_0000_0000_0000 (bits 47–61).
  - Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62).
  - Preferred Stop value is 0.

Refer to the Tamper Detect section of DCF client list table (*Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*) for details.

Note: User software must check whether the diary is full. If the diary area for a TDR is full, user software must program a DCF record to override blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry. This is detailed in Section 36.6.4: Overriding tamper detection.

36.6.3 Assigning blocks to Tamper Detection Regions (TDRs)

Tamper detection regions are defined by creating TDR*x*_LOCK*n* DCF records. For example, in a microcontroller with six tamper detection regions, the following records define the TDRs:

- TDR0_LOCK0, TDR0_LOCK1, TDR0_LOCK2, TDR0_LOCK3
- TDR1_LOCK0, TDR1_LOCK1, TDR1_LOCK2, TDR1_LOCK3
- TDR2_LOCK0, TDR2_LOCK1, TDR2_LOCK2, TDR2_LOCK3
- TDR3_LOCK0, TDR3_LOCK1, TDR3_LOCK2, TDR3_LOCK3
- TDR4_LOCK0, TDR4_LOCK1, TDR4_LOCK2, TDR4_LOCK3
- TDR5_LOCK0, TDR5_LOCK1, TDR5_LOCK2, TDR5_LOCK3

Each set of TDRxLOCK*n* DCF records mapped in exactly the same way as the Flash module's SEL*n* and LOCK*n* registers (refer to the detailed Flash block map in the Flash section of the Device Configuration chapter for the mapping).

In the TDRxLOCK*n* DCF records:

- A bit set to '1' means the corresponding Flash block is assigned to this TDR.
- A bit set to '0' means the corresponding Flash block is NOT assigned to this TDR.

The default state (no TDRs defined) is defined as "Block NOT assigned to the TDR".

36.6.4 Overriding tamper detection

When a TDR's diary section is full, a Tamper Override (TO) record must be created before a block assigned to the TDR can be programmed or erased.

The details of the data section of the DCF record used to override tamper detection is shown in *Figure 373*.

Caution: Bits in this DCF record are writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0

Figure 373. Data portion of Tamper Region Override (TO) DCF record

Example 8 Implementing secure write protection

Assume the diary Tamper Detection Region 3 is full.

We need to create one DCF record that defines that overrides tamper detection for the tamper detection region whose diary is full. The information required is from the Tamper Detect section of *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

Table 461. DCF client list of Tamper Detect

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0001	Tamper Region Override

Referring to *Figure 369*, the values for the DCF record are:

- Data[31:0] is 0x0000_0008 (TOE3).
- CS[14:0] is 0b000_0000_0001_0000 (bits 32–46).
- Address[16:2] is 0b000_0000_0000_0001 (bits 47–61).
- Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
- Preferred Stop value is 0.

Refer to the Tamper Detect section of DCF client list table (*Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*) for details.

36.7 Implementing OTP

Any Flash block within the Flash array can be assigned, at any time, to be OTP. OTP means that Flash erase of the entire block is disabled and only 128-bit quad-words that are already erased (Flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block becomes OTP after the next reset.

Caution: After a Flash block is configured as OTP, it cannot be changed back.

This capability resides within the Tamper Detection Module (TDM). The records are detailed in *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*. Refer to the entries for the OTP_EN0, OTP_EN1, OTP_EN2, and OTP_EN3 DCF records in the Tamper Detect section of the table.

36.8 Implementing test mode disable

The SPC584Bx Flash memory module includes a mechanism to disable manufacturer entry into test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method are able to have possible failures analyzed by manufacturer's failure analysts.

Test mode disable prevents all high voltage operations to the Flash executed by the internal state-machine, as well as reads through the state machine, and reads through the Array Integrity state machine when using test mode interfaces.

Optionally, the customer can create a password to enable manufacturer entry into test mode.

36.8.1 Unconditional test mode disable seal

Warning: This process is not reversible. After completing these steps the SPC584Bx microcontroller is permanently prevented from entering test mode for factory analysis.

The steps for unconditionally disabling test mode are the same steps required for implementing passcode-protected entry into test mode with one exception—the passcode specified must be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To unconditionally disable test mode entry:

1. Select individual Flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST Flash memory map.
2. Program one of the three *invalid* passcodes shown above into the Test Mode Override Passcode area shown in the UTEST Flash memory map.
3. Program the value 0x5A4B_3C2D into the Test Mode Disable Seal location specified in the UTEST Flash memory map.

After the next reset is asserted, Test Mode is permanently disabled.

Note: Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.

Note: The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

36.8.2 Passcode-protected test mode disable seal

The steps for implementing passcode-protected entry into test mode are the same required for unconditionally disabling test mode with the exception that —the passcode specified must NOT be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To implement passcode-protected test mode entry:

1. Select individual Flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST Flash memory map.
2. Program a valid passcode into the Test Mode Override Passcode area shown in the UTEST Flash memory map.
3. Program the value 0x5A4B_3C2D into the Test Mode Disable Seal location specified in the UTEST Flash memory map.

After the next reset is asserted, Test Mode is permanently passcode protected.

Note: Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.

Note: The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

36.8.3 Selecting Flash memory blocks for test mode disable seal

Selecting Flash memory blocks for test mode disable seal protection is similar to selecting blocks for tamper detection in that it involves writing to non-register Flash locations. In the case of the test mode disable seal function, there are two areas of UTEST Flash memory that have identical mapping to Flash memory blocks. When the test mode disable seal takes effect, the mapped bits are logically ORed to determine the blocks to be protected.

The mapping of bits to blocks for this function is identical to the mapping used for other functions. Refer to *Chapter 35: Embedded Flash Memory* for the map.

Note: Selecting a block for test mode disable requires a '0' in the mapped bit—not a '1' as with other functions.

36.9 Security configuration planning

At minimum, designing the Flash security framework consists of selecting the security features to be implemented and determining the scope of implementation for each selected feature.

In devices containing a Hardware Security Module (HSM), significant planning must go into the implementation of HSM functions. The HSM is an independent software platform within the microcontroller. It is intended to be used to implement advanced security functions that are defined by customer software.

Note: Details of advanced security features, including the advanced security architecture is and the Hardware Security Module (HSM) are contained in the SPC584Bx Microcontroller Security Reference Manual. Distribution of the document is restricted to qualified customers.

The basic security features available in the SPC584Bx microcontroller include:

- Secure write protection
- Secure read protection
- Debug port enable/disable
- One-time-programmable (OTP) Flash memory block assignment
- Factory test mode disable

36.9.1 Hardware Security Module (HSM)

Details of the HSM are included in the *SPC584Bx Microcontroller Security Reference Manual*. It is mentioned here because the Flash memory module includes blocks that can be configured for exclusive access by the HSM. If the HSM is to be used in an application it is advisable to protect HSM code by configuring it for exclusive access by the HSM. The blocks configurable for exclusive access by the HSM appear in *Figure 5* in *Chapter 3: Embedded memories*. They are labeled as secure code Flash or secure EEPROM emulation blocks. They are not configured for HSM-exclusive access by default, though—they are available to act as secure Flash areas but must be explicitly configured for that purpose. Refer to *SPC584Bx Microcontroller Security Reference Manual* for details.

36.9.2 Creating password groups

Most basic security features in the SPC584Bx microcontroller are password-based. Therefore, one of the early tasks that must be performed is the creation of four 256-bit passwords. They are programmed by the customer into the locations identified in the UTEST memory map (refer to the Memory Map chapter).

The UTEST memory map fields to be written are:

- PASS Password Group 0
- PASS Password Group 1
- PASS Password Group 2
- PASS Password Group 3

Passwords must not be one of the following, which are considered insecure and therefore invalid:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

Caution: The password fields are in OTP UTEST Flash, so they can only be written once.

36.9.3 Planning secure write protection

Up to four levels of password-secured write protection can be assigned to any block. Note that by default, all blocks of user Flash are write protected, so this could be more accurately viewed as planning which blocks of user Flash will *not* have four levels of password-secured write protection. The write protection desired will depend on the usage of the Flash blocks. Before creating DCF records to implement secure write protection, please thoroughly review the implementation steps detailed in *Section 36.3.1: Implementing secure write protection* and *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

36.9.4 Planning secure read protection

Up to four levels of password-secured write protection can be assigned to Flash blocks in pre-defined Read Locking groups. Note that by default, all blocks of user Flash are read protected, so this could be more accurately viewed as planning which blocks of user Flash will *not* have four levels of password-secured read protection. The read protection desired will depend on the usage of the Flash blocks. Before creating DCF records to implement secure read protection, please thoroughly review the implementation steps detailed in *Section 36.4.1: Implementing secure read protection* and *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

36.9.5 Planning debug port enable/disable

Refer to *Section 36.5: Debug port enable/disable* and *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

36.9.6 Planning OTP Flash memory block assignment

Any block of user Flash can be configured as OTP. The need to configure blocks as OTP will depend on the data or code to be stored in the block. It is important to note that configuring a Flash block as OTP is not reversible.

Before creating DCF records to implemented OTP Flash blocks, please thoroughly review the implementation steps detailed in *Section 36.7: Implementing OTP* and *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

36.9.7 Planning factory test mode disable

Refer to *Section 36.8: Implementing test mode disable* and *Table 107: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

37 Decorated Storage Memory Controller (DSMC)

37.1 Introduction

This section details hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called "*decorated storage*". It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A "decoration" is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l[b,h,w]d{cb}x rT,rA,rB` where the data size specifier is defined as 8-bit (b = byte), 16-bit (h = halfword) or 32-bit (w = word), and the three register specifiers include `rT` as the destination target register, `rB` as the effective address and `rA` as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st[b,h,w]d{cb}x rS,rA,rB` where the same data size specifiers are used, and the three register specifiers are `rS` as the source data register, `rB` as the effective address and `rA` as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the rB register (hence the `"x"` mnemonic suffix, signaling an "indexed" addressing mode) and the decoration is specified by the rA register. The core transmits the contents of the rA register as the decoration value along with the access address (register rB) to the DSMC.

The decorated load and store instruction formats are shown in *Figure 374*.

Figure 374. Power Architecture Decorated Load and Store Instruction Formats

	0	1	2	3	4	5	6–10	11–15	16–20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1	**r**T	**r**A	**r**B	1	0	0	0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1	**r**T	**r**A	**r**B	1	0	0	0	1	0	0	0	1	1	/
lwdx	0	1	1	1	1	1	**r**T	**r**A	**r**B	1	0	0	1	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1	**r**T	**r**A	**r**B	1	0	0	0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1	**r**T	**r**A	**r**B	1	0	0	0	1	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1	**r**T	**r**A	**r**B	1	0	0	1	0	0	0	0	1	0	/

	0	1	2	3	4	5	6–10	11–15	16–20	21	22	23	24	25	26	27	28	29	30	31
stbdx	0	1	1	1	1	1	**r**S	**r**A	**r**B	1	0	1	0	0	0	0	0	1	1	/
sthdx	0	1	1	1	1	1	**r**S	**r**A	**r**B	1	0	1	0	1	0	0	0	1	1	/
stwdx	0	1	1	1	1	1	**r**S	**r**A	**r**B	1	0	1	1	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1	**r**S	**r**A	**r**B	1	0	1	0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1	**r**S	**r**A	**r**B	1	0	1	0	1	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1	**r**S	**r**A	**r**B	1	0	1	1	0	0	0	0	1	0	/

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit “decoration”. The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and operand data for the command or both.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (`l[b,h,w]dcbx`) and store (`st[b,h,w]dcbx`) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

37.2 Decorated Stores: `st[b,h,w]d{cb}x rS,rB,rA`

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a `<command>.<size>` syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
 8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
 8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- wdata = rS

37.2.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit Memory Container

Figure 375. Decoration Format: BFINS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bfins.b	0	0	0	0	0	0	0	SRTBIT			0	0	0	BFW			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.h	0	0	0	0	0	0	SRTBIT				0	0	BFW				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.w	0	0	0	0	0	SRTBIT					0	BFW					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command inserts a bit field defined by SRTBIT and BFW into the memory "container" defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read

if bfw == 0
   then bfw = container

if ((srtbit + bfw) <= container)// generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp   & ~mask// modify
    | wdata &  mask

mem[accessAddress, size] = tmp// memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd_efgh". For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz-_----,
   then destination is "xyzd_efgh"
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
   then destination is "axyz_efgh"
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
   then destination is "abxy_zfgh"
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
   then destination is "abcx_yzgh"
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
   then destination is "abcd_xyzh"
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_-xyz,
   then destination is "abcd_exyz"
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
   then destination is "abcd_efxy"
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
   then destination is "abcd_efgx"
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

37.2.2 Compare-and-Store (CAST)

Figure 376. Decoration Format: CAST

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cast.b	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CD8							
cast.h	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	CD16															
cast.w	1	0	0	1	CD28																											

This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size] // memory read

if (size == 8)// define compare_operand
    compare_operand = CD8
else if (size == 16)
          compare_operand = CD16
     else compare_operand = {0x0, CD28}// zero-filled data

if (tmp == compare_operand)// compare - "modify"
     mem[accessAddress, size] = wdata// memory write
else mem[accessAddress, size] = tmp
```

Note for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit compare operand.

37.2.3 Logical AND (AND)

Figure 377. Decoration Format: AND

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
and.{b,h,w}	1	0	1	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata// modify
mem[accessAddress, size] = tmp// memory write
```

37.2.4 Logical OR (OR)

Figure 378. Decoration Format: OR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
or.{b,h,w}	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata// modify
mem[accessAddress, size] = tmp// memory write
```

37.2.5 Logical Exclusive-OR (XOR)

Figure 379. Decoration Format: XOR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
xor.{b,h,w}	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata// modify
mem[accessAddress, size] = tmp// memory write
```

37.3 Decorated Loads: `l[b,h,w]d{cb}x rT,rB,rA`

The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an “extract”) operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

37.3.1 Simple Load (SLD)

Figure 380. Decoration Format: SLD

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sld.{b,h,w}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

37.3.2 Registers-and-Memory Exchange (SWAP)

Figure 381. Decoration Format: SWAP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cast.b	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	WD8							
cast.h	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	WD16															
cast.w	0	1	0	1	WD28																											

This command loads the contents of the referenced memory location into the core’s rT register and stores the source operand from the core’s rA register (the decoration) into the memory location.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp

if (size == 8)// store_operand - "modify"
    store_operand = WD8
else if (size == 16)
          store_operand = WD16
     else store_operand = {0x0, WD28}// zero-filled write data

mem[accessAddress, size] = store_operand// memory write
```

Note for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32-bit store operand.

37.3.3 Load-and-Set-1(Bit) (LAS1)

Figure 382. Decoration Format: LAS1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	0	BIT			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp
mask  = 1 << (container - bit - 1)// generate bit mask
tmp   = tmp | mask// modify
mem[accessAddress, size] = tmp// memory write
```

37.4 DSMC Instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the data port of the cores and the master ports of the crossbar.

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated memory controller only operates on aligned, single data transfers (misalignment and bursts are not supported and error terminated if attempted).

As the DSMC generates the atomic read-modify-write bus transactions to the targeted slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

38 Analog-to-Digital Converters (ADC) Configuration

This chapter is organized as follows:

- *Section 38.1: ADC overview*
  - *Section 38.1.1: ADC subsystem block diagram*
  - *Section 38.1.2: Analog input pin multiplexing*
- *Section 38.2: Configuration of ADC modules*
  - *Section 38.2.1: Successive Approximation Register Analog-to-Digital Converter (SAR ADC)*

38.1 ADC overview

SPC584Bx contains the following analog-to-digital converters (ADCs):

- three independent 12-bit and one independent 10-bit Successive Approximation Register Analog-to-Digital Converters (SARADCs)

Note: SARADC instances naming in some figures of this chapter may differ from the naming provided in Table 462 in order to simplify these figures.

Table 462. ADC implementation

Converter Type	12-bit SAR ADC			10-bit SAR ADC
Special Function	**Supervisor SAR ADC**	**Fast SAR ADC**		**STDBY & Fast Comparator**
Name	SAR_ADC_12bit_SUPERVISOR (SAR_ADC_B)	SAR_ADC_12bit_0	SAR_ADC_12bit_1	SAR_ADC_10bit_STDBY
Number of internal channels(1)	63	11	11	9

1. This is the maximum number of available channels per dedicated silicon but the available number of channels per package might be different (see the IO_Definition document)

The internal channels of each ADC are connected to analog input pads. The analog pads include programmable pull-up / pull-down resistors.

These ADCs interface to external analog input pins. The number of external analog input pins for each package is available in the IO_Definition.xls file attached to the IO_Definition document.

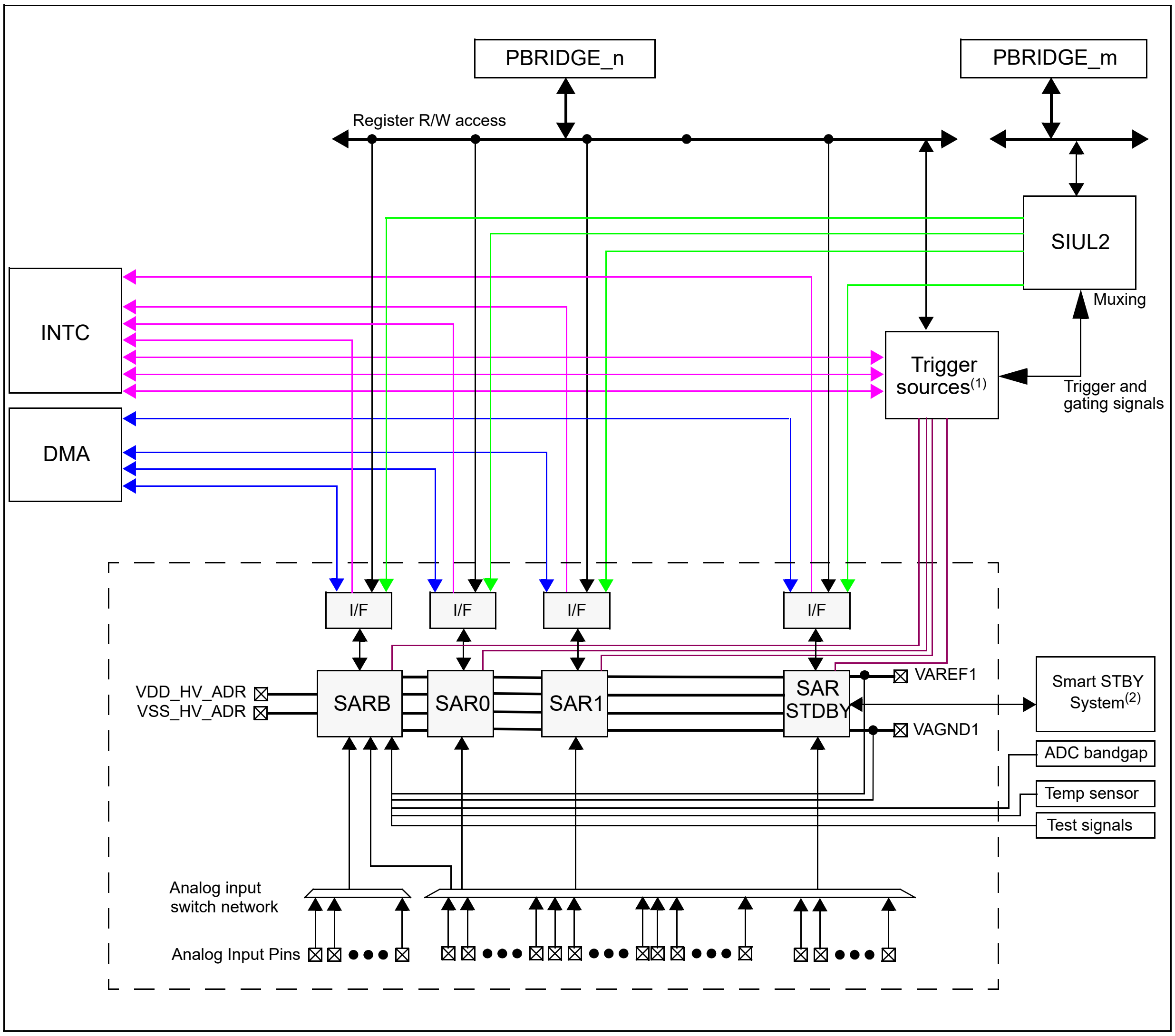
Each SARADC instance has independent analog input, register interface, trigger and clock inputs, independent interrupt, and DMA capability. Each SARADC has a dedicated peripheral bridge (PBRIDGE) interface for CPU access to the registers.

All SARADCs share the same supply pins (VDD_HV_ADV/VSS_HV_ADV). This is true for all packages.

38.1.1 ADC subsystem block diagram

Figure 383 gives the block diagram for the high-level integration of the ADC sub-system for SPC584Bx. The primary intent of the diagram is to show that each ADC has independent register interfaces, interrupt/DMA requests, and trigger source selection.

Figure 383. SPC584Bx ADC subsystem



1. The different trigger sources are detailed in the *Section 38.2.1.5: SAR ADC triggering*
2. SAR_ADC_10bit_STDBY interactions with the Smart STBY System are described in the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)*

38.1.2 Analog input pin multiplexing

For analog input pin multiplexing please refer to the IO_Definition.xls file attached to the IO_Definition document.

38.2 Configuration of ADC modules

The following sections describe device-specific details for the following module:

- *Section 38.2.1: Successive Approximation Register Analog-to-Digital Converter (SAR ADC)*

38.2.1 Successive Approximation Register Analog-to-Digital Converter (SAR ADC)

There are three independent 12-bit and one independent 10-bit SAR ADCs on SPC584Bx. Refer to *Figure 383* for the ADC sub-system block diagram.

The following sections describe configuration details:

- *Section 38.2.1.1: SAR ADC configuration overview*
- *Section 38.2.1.2: SAR ADC integration diagram*
- *Section 38.2.1.3: SAR ADC analog input*
- *Section 38.2.1.4: SAR ADC clock sources*
- *Section 38.2.1.5: SAR ADC triggering*
- *Section 38.2.1.6: SAR ADC Standby*
- *Section 38.2.1.7: SAR absolute voltage reference*

For detailed module information, refer to the Successive Approximation Register Analog-to-Digital Converter (SAR ADC) Digital Interface chapter.

38.2.1.1 SAR ADC configuration overview

38.2.1.1.1 12-bit Fast SAR ADC

Two 12-bit Fast SAR ADC instances are implemented with:

- 12-bit and 10-bit conversion mode selectable by software on a per conversion basis
  - 11 analog input channels for each SAR_ADC_12bit_0, SAR_ADC_12bit_1
- 4 watchdogs

38.2.1.1.2 12-bit SAR ADC B0

One 12-bit SARADC B0 (SAR_ADC_12BIT_B0) instance with diagnosis is implemented with:

- 12-bit and 10-bit conversion mode selectable by software on a per conversion basis
- 63 analog input channels (called internal channels)
- 64 external multiplexed channels (called external channels)
- All analog internal channels from other ADCs are also connected to SAR_ADC_12bit_B0
- 9 internal test channels (called "test channels") to be used for self-diagnosis of the converter and diagnosis of the analog signal path up to the external circuitry. This includes:
  - an internal bandgap reference from temperature sensor
  - an internal analog ground signal for software offset cancellation
  - four internal voltage references with 20 kOhm source impedance
- 8 watchdogs
- Body Cross Triggering Unit (BCTU) support to trigger ADC conversion

38.2.1.1.3 10-bit Fast SAR ADC STDBY

One fast 10-bit SAR ADC STDBY instance with 9 internal channels with fast comparator capability and supporting STAND-BY feature is implemented with:

- 10-bit and 8-bit conversion mode selectable by software on a per conversion basis (all standard SAR ADC functionalities are present)
- 8 external channels needed for data register and watchdog functionality
- 4 analog watchdogs
- Cross Triggering Unit (STBY eCTU) support. In Standby mode, ADC operation is controlled by STBY eCTU. Refer to the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* for details on STAND-BY feature
- In RUNTIME, 10-bit Fast ADC STDBY can be used as the other 10-bit Fast SAR ADC instances

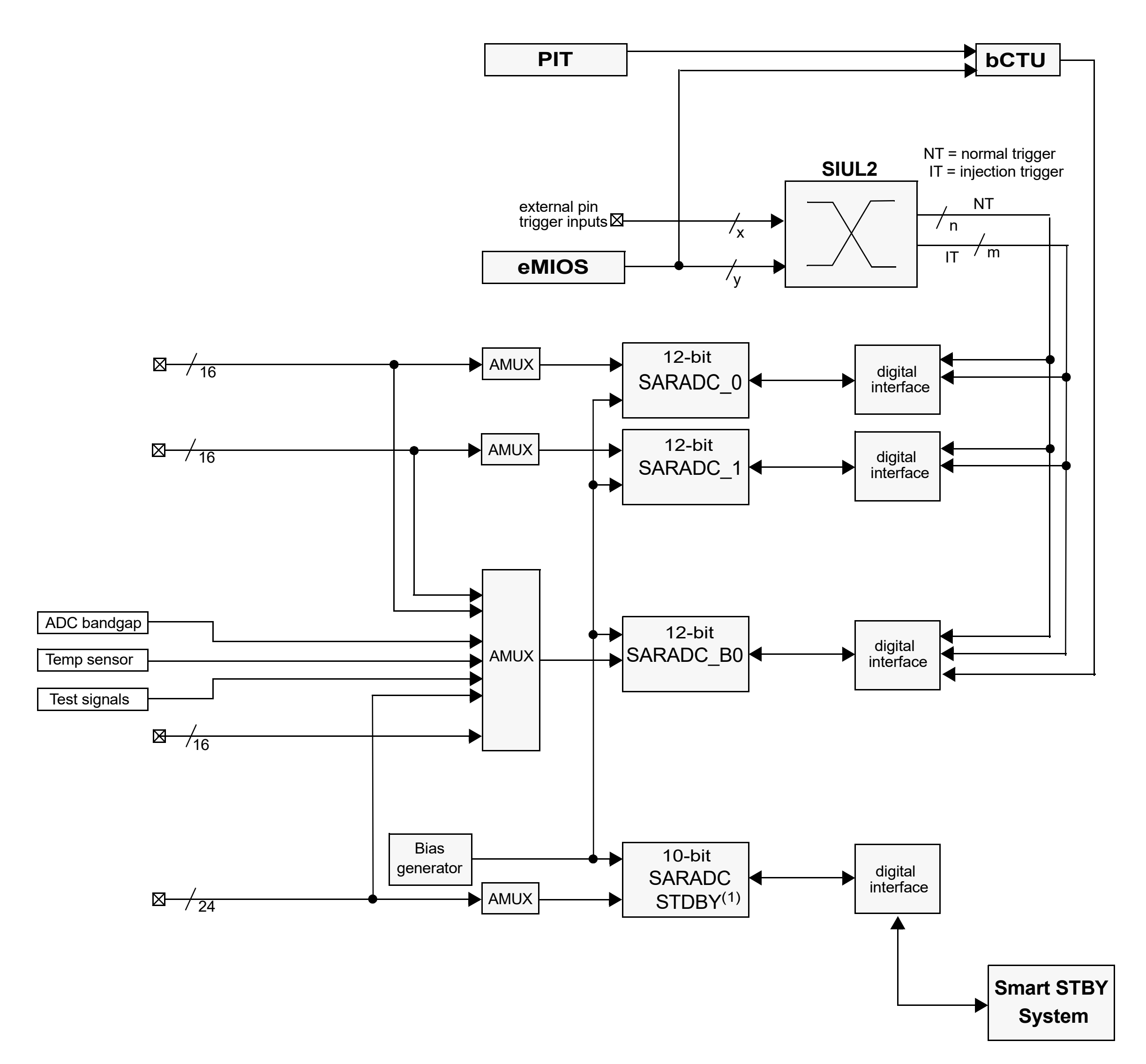
Note: Refer to SPC584Bx datasheet for SAR ADC analog performances.

38.2.1.2 SAR ADC integration diagram

Figure 384 shows the block diagram of the integration of the four 12-bit and one 10-bit SAR ADCs on SPC584Bx.

The bias generator provides 4 reference points (VDD_HV_ADR_S, 1/3 VDD_HV_ADR_S, 2/3 VDD_HV_ADR_S, and VSS_HV_ADR_S) which are provided to SAR_ADC_12bit_B0 via an internal 20 kΩ impedance. See the SAR ADC block diagram in the *Chapter 39: Successive Approximation Register Analog-to-Digital Converter (SARADC)*.

Figure 384. SAR ADC integration diagram(1)(2)



1. SAR_ADC_10bit_STDBY instance is implemented on a special low-power domain. Further information is provided in the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)*
2. No normal and injection trigger for SAR_ADC_STDBY

38.2.1.3 SAR ADC analog input

All analog input pins are routed to SAR_ADC_12bit_B0 and another fast SAR ADC Analog input (AMUX) is implemented in the SAR ADC subsystem.

Note: Simultaneous sampling of two SAR ADCs on the same analog channel is possible under certain conditions:

Both ADCs need synchronized start of sampling (both ADCs have to be triggered by the same trigger signal)

Both ADCs need synchronized end of sampling (both ADCs have to be configured with the same sampling time)

External circuitry on the analog input has to be chosen accordingly (effective internal capacity on analog signal path during sampling time is doubled versus single conversion as given in the device data sheet)

The mapping of analog input pins to the SAR converters is given in *Section 38.1.2*.

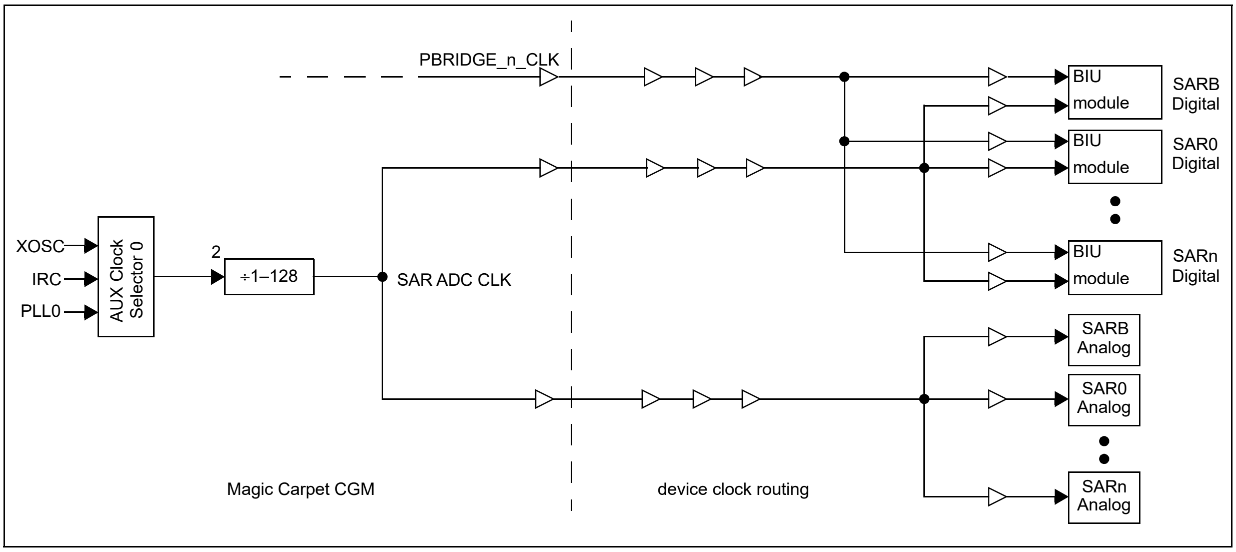
38.2.1.4 SAR ADC clock sources

Each SAR ADC digital interface on SPC584Bx has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE*n*_CLK. The module clock is sourced by the ADC clock (SAR ADC CLK). See *Chapter 25: Clocking* for more detail on the clock sources.

The input clock for the SAR ADC analog blocks is clocked by the SAR ADC CLK. *Figure 385* shows a block diagram of the SAR ADC clock architecture.

To avoid crosstalk issues, all SAR ADCs run at the same clock frequency. The SAR digital interface is clocked from the PBRIDGE*n*_CLK clock. All SAR analog blocks are clocked from the output of a single divider of the SAR ADC CLK at the device level.

Figure 385. SAR ADC clock diagram



38.2.1.5 SAR ADC triggering

Each SAR ADC has 2 independent trigger inputs for starting normal conversion or injected conversion on an analog input. One trigger source can be selected among a list of available signals and connected to the trigger inputs independently for each SAR ADC.

The trigger sources for each SAR ADC on the device are either external input pins or eMIOS channels. Each external input can selectively trigger any of the SAR ADCs on the device. The synchronization to the PBRIDGE clock and digital filtering of the external input pin triggers is done in the system integration logic.

Each trigger input supports software selectable options for rising edge, falling edge, rising and falling edge, or level-based trigger assertion. The trigger selection for each SAR ADC and edge/level selection is located in the SIUL2.

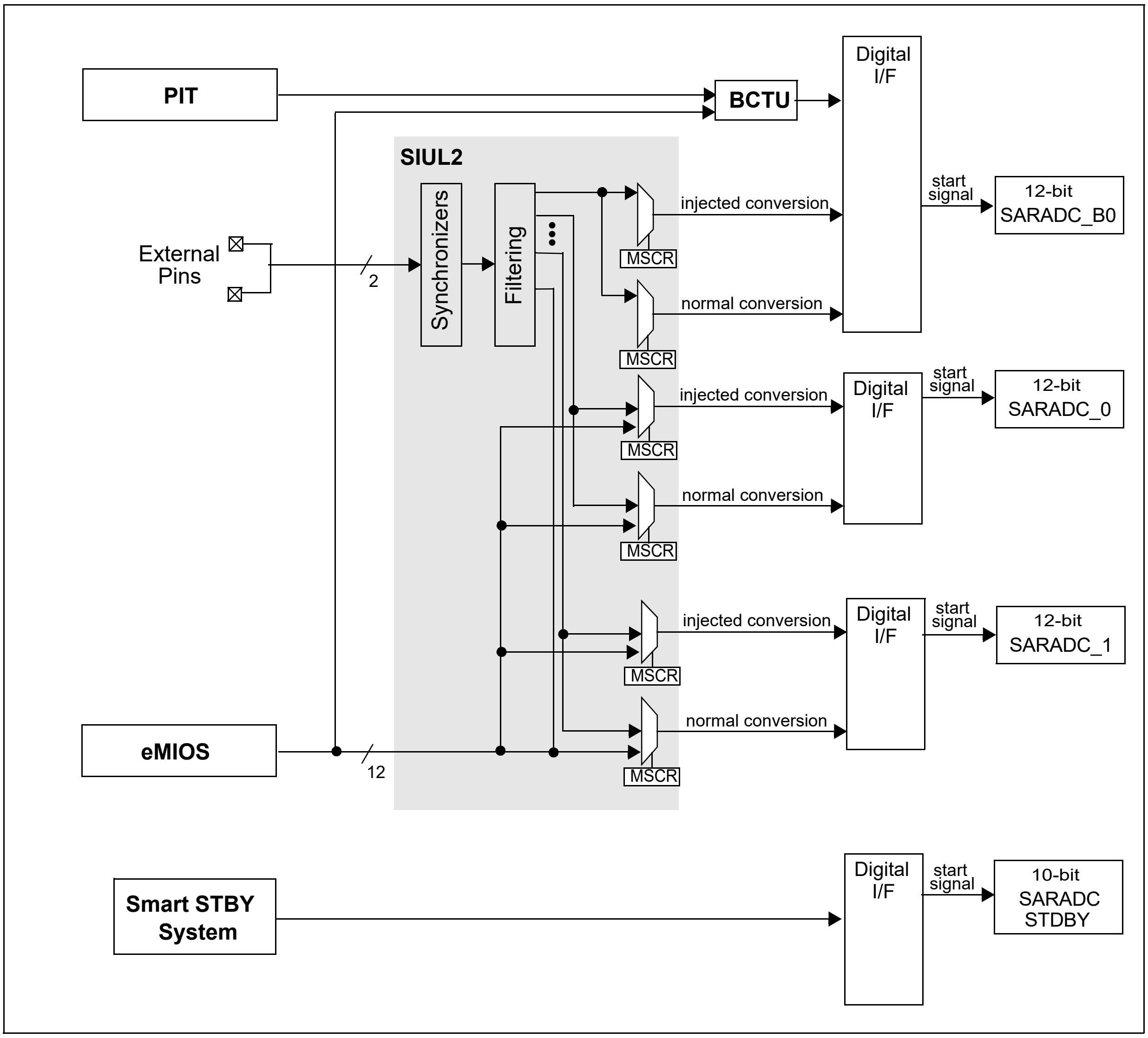
The BCTU module is connected to SAR_ADC_12bit_B0 (SAR_ADC parameter CTSEN = 0). This module receives trigger requests from the timer module (PIT_RTI)or from EMIOS. It translates those trigger requests into ADC measurement commands.

The STBY eCTU module is connected to SAR_ADC_10bit_STDBY (SAR_ADC parameter CTSEN = 2). It receives trigger requests from the real time clock module (RTC) and translates them into ADC commands. The STBY eCTU is used for the Smart Standby feature. Please refer to the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* for more data about this feature.

The SPC584Bx SoC diagram for the SAR ADC trigger inputs is given in *Figure 386*.

Note: Please refer to the MS Excel file attached to SPC584Bx IO_Definition document (I/O signal table) to get the exhaustive list of signals source able to trigger ADC conversion.

Figure 386. SAR ADC input trigger diagram(1)



1. No normal and injection triggers for SARADC_STDBY

Trigger source selection for each SAR ADC is given in the *Chapter 16: System Integration Unit Lite2 (SIUL2)*.

Each SAR ADC digital interface, except SAR_ADC_12bit_B0, provides 4 watchdog Threshold Crossover event trigger signals which are mapped to the interrupt controller.*Chapter 16: System Integration Unit Lite2 (SIUL2)*

38.2.1.6 SAR ADC Standby

SAR_ADC_10bit_STDBY interactions with the Smart STBY System are described in the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)*

38.2.1.7 SAR absolute voltage reference

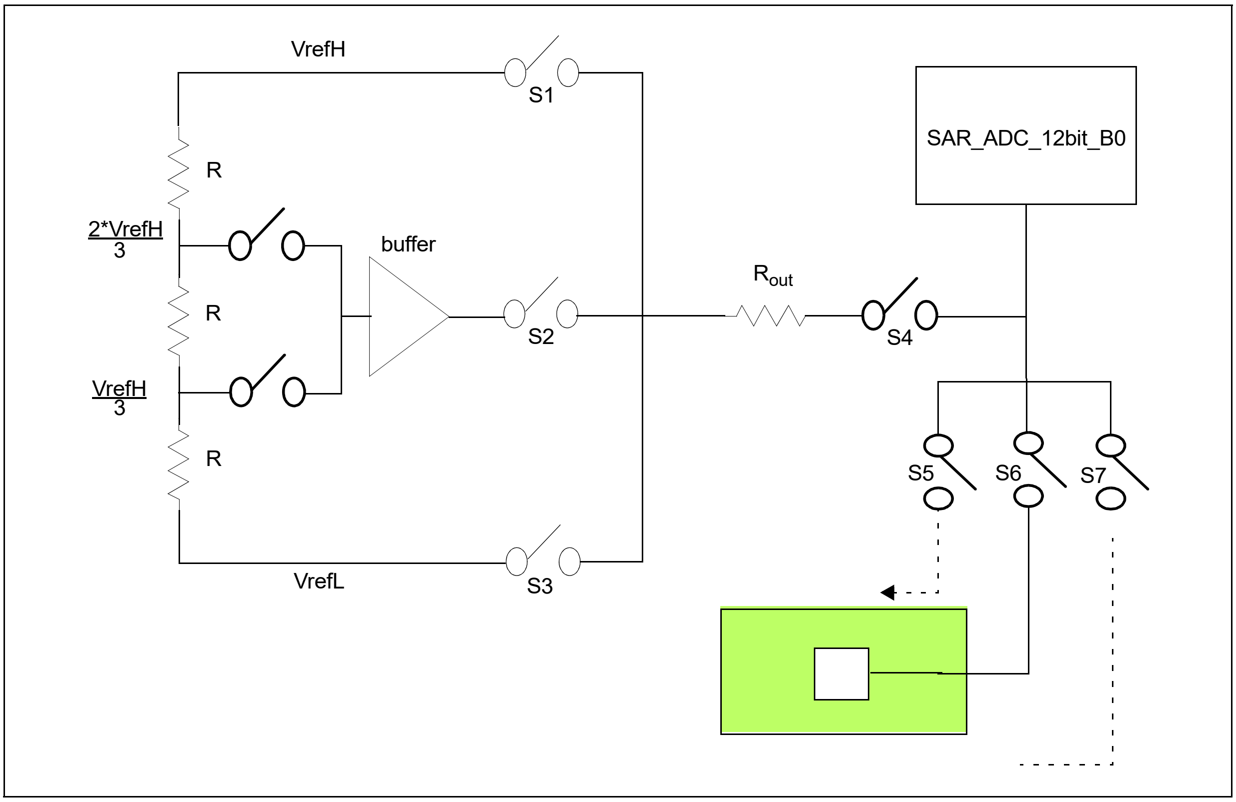
SPC584Bx provides an internal, absolute voltage reference that can be calculated from two digital temperature sensor signal values. The temperature sensor signal values can be read by SAR_ADC_12bit_B0. For further information refer to *Chapter 40: Temperature Sensor*. The ADC conversion channel numbers for the temperature sensor signals are given in *Table 464*.

38.2.1.8 SAR ADC diagnostic

38.2.1.8.1 Self Test features

- The ADCBIAS provides four different voltage levels defined as GND, $V_{ref}/3$, $2*V_{ref}/3$ and V_{ref}. The self test internal reference voltage precision is provided in the SAR ADC electrical specification table of the device datasheet.
- For the impedance requirements please refer to SPC584Bx datasheet.
- The impedance of the switches that are connecting the different references to the ADC input has not been considered in the impedance calculation.
- The impedance of the switches from the reference generation to the ADC input has been matched with the one of the switches that are connecting the device pin to the ADC input.
- The sampling time of SAR_ADC_12bit_B0 needs to be extended due to long settling time induced by the high impedance of the source. For sampling time adjustment, please refer to SPC584Bx datasheet.
- The SAR ADC Digital Interface provides the necessary control to select the required switches S1–S7 (shown in *Figure 387*) to perform the Self Test.

Figure 387. Self Test implementation



38.2.1.8.2 Internal reference

SAR_ADC_12bit_B0 provides the ability to sample and convert four internal voltages through a 20 kΩ source impedance. These voltages are as follows:

- $V_{DD\_HV\_ADR\_S}$
- 1/3 $V_{DD\_HV\_ADR\_S}$
- 2/3 $V_{DD\_HV\_ADR\_S}$
- $V_{SS\_HV\_ADR\_S}$

Conversion of an internal reference voltage works the same as that for a normal conversion, with the exception of the conversion time. Due to the 20 kΩ source impedance, the sampling time is much higher than the normal time.

Enabling the sampling and selection of an internal reference is independent of input channel selection for SAR_ADC_12bit_B0.

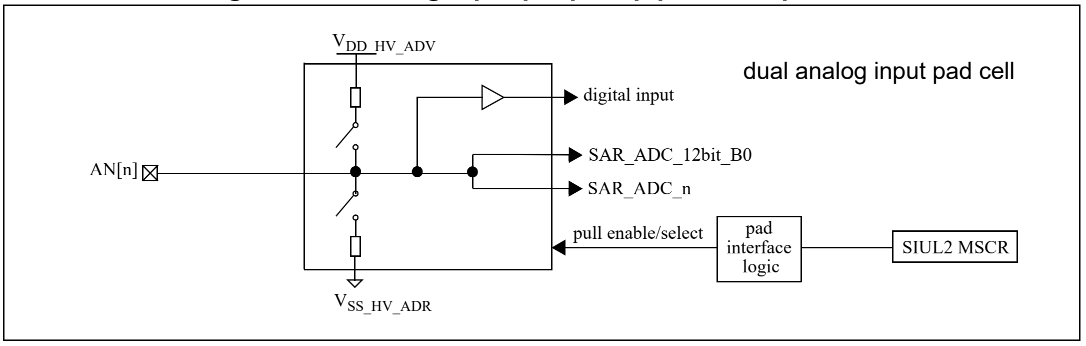
38.2.1.8.3 Analog input pin programmable pullup/pulldown

An open or short circuit condition is detectable within the device for analog input pins. There is a programmable pullup/pulldown pad cell at the input pad. This allows for application testing of the mux input logic. The analog input pin diagram with the pullup/pulldown is shown in *Figure 388*. The pullup/pulldown is included inside the analog input pad cell.

Control of the pullup/pulldown is independently done in the SIUL2 MSCR registers for each analog input pin.

Only the analog input pins specified in *Section 38.1.2* have the pullup/pulldown feature.

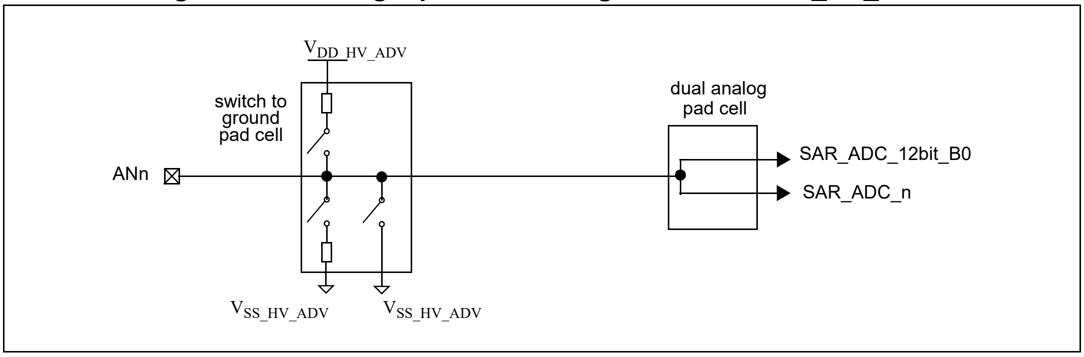
Figure 388. Analog input pin pullup/pulldown pad cell



38.2.1.8.4 Analog input pin switch to $V_{SS\_HV\_ADV}$

All analog input-only pins on SPC584Bx have in addition to the weak pullup/pulldown a programmable switch to $V_{SS\_HV\_ADV}$ to discharge the input. The analog input diagram is given in *Figure 389*. The default state of the switch for both inputs is OFF.

Figure 389. Analog input — discharge switch to VSS_HV_ADV



The ground switches are controlled independently from the input switch to the ADC. The ground switch is controlled in input multiplexing MSCR registers (MSCR[WPDE], MSCR[WPUE]) in the SIUL2. The MSCR register provides selection between open or closed switch, with open being the default state.

The ground switches can be used to periodically discharge the associated channels.

38.2.1.9 SAR ADC channel assignment

38.2.1.9.1 Internal channel assignment

The input multiplexer channel assignments for the SAR ADCs are provided in *Table 463*.

Table 463. SAR analog input channel assignment

Analog input pin	SAR_ADC_12bit_B0 input channel	Fast SAR	Fast SAR channel	Description
AN[11:0]	11–0	—	—	Analog input pin
AN[13:12]	13–12	SAR_ADC_12bit_1	13–12	Analog input pin
AN[14]	14	—	—	Analog input pin
AN[17:15]	17–15	SAR_ADC_12bit_1	17–15	Analog input pin
AN[26:18]	26–18	—	—	Analog input pin
AN[31:27]	31–27	SAR_ADC_12bit_1	31–27	Analog input pin
AN[32]	32	—	—	Analog input pin
AN[33]	33	SAR_ADC_12bit_1	33	Analog input pin
AN[35:34]	35–34	SAR_ADC_12bit_0	35–34	Analog input pin
AN[37:36]	37–36	—	—	Analog input pin
AN[42:38]	42–38	SAR_ADC_12bit_0	42–38	Analog input pin
AN[43]	43	—	—	Analog input pin
AN[44]	44	SAR_ADC_12bit_0	44	Analog input pin
AN[46:45]	46–45	—	—	Analog input pin
AN[49:47]	49–47	SAR_ADC_12bit_0	49–47	Analog input pin
AN[65:50]	65–50	SAR_ADC_12bit_3	65–50	Analog input pin
AN[71:66]	71–66	—	—	Analog input pin

Table 463. SAR analog input channel assignment (continued)

Analog input pin	SAR_ADC_12bit_B0 input channel	Fast SAR	Fast SAR channel	Description
AN[73:72]	73–72	SAR_ADC_10bit_STDBY	73–72	Analog input pin
AN[74]	74	—	—	Analog input pin
AN[77:75]	77–75	SAR_ADC_10bit_STDBY	77–75	Analog input pin
AN[78]	78	—	—	Analog input pin
AN[82:79]	82–79	SAR_ADC_10bit_STDBY	82–79	Analog input pin
AN[94:83]	94–83	—	—	Analog input pin

38.2.1.9.2 Test channel assignments

Each test channel has a unique channel assignment, and is treated by software like any other A/D conversion. Access to the test channels is required for SAR_ADC_12bit_B0 only. The test channel assignments for SAR_ADC_12bit_B0 are given in *Table 464*.

Table 464. SAR_ADC analog test channel assignment

SAR_ADC_12bit_B0 input channel	Description
96	Reserved
97	Reserved
98	Reserved
99	Reserved
100	Reserved
101	Reserved
102	Reserved
103	Reserved
104	VDD_LV
105	Reserved
106	Reserved
107	Reserved
108	Reserved
109	VSS_HV_ADV (Ground Supply for ADC)
110	Reserved
111	Reserved
112	Reserved
113	Reserved
114	Reserved
115	Reserved

Table 464. SAR_ADC analog test channel assignment (continued)

SAR_ADC_12bit_B0 input channel	Description
116	Reserved
117	Reserved
118	Reserved
119	Reserved
120	PTAT (Proportional To Absolute Temperature)
121	Temperature sensor CTAT (Complementary To Absolute Temperature)
122	VSS_LV
123	Reserved
124	SAR BIAS 0 — VSS_HV_ADR_S through 20 KΩ source impedance
125	SAR BIAS 1 — 1/3 (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance
126	SAR BIAS 2 — 2/3 (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance
127	SAR BIAS 3 — (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance

39 Successive Approximation Register Analog-to-Digital Converter (SARADC)

39.1 Introduction

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for the application. It provides accurate and fast conversion data for a wide range of applications. Each SARADC analog block has its corresponding digital interface implemented at the chip level.

39.2 Overview

The SARADC digital interface contains advanced features for normal or injected conversion modes of operation. The conversion can be triggered by software or hardware. For advanced cross triggering functions, it provides a CTU interface with an on-chip Cross Triggering Unit (CTU) which can support two different CTU modes: CTU “trigger mode” and CTU “control mode” (refer to details in *Section 39.4.5: Cross Triggering Unit (CTU) interface*).

The digital interface can be configured to control up to 256 multiplexed analog input channels. There are three types of input channels:

- Internal channels, mapped to index ranging from 0 to 95.
- Test channels, mapped to index ranging from 96 to 127.
- External channels, mapped to index ranging from 128 to 255.

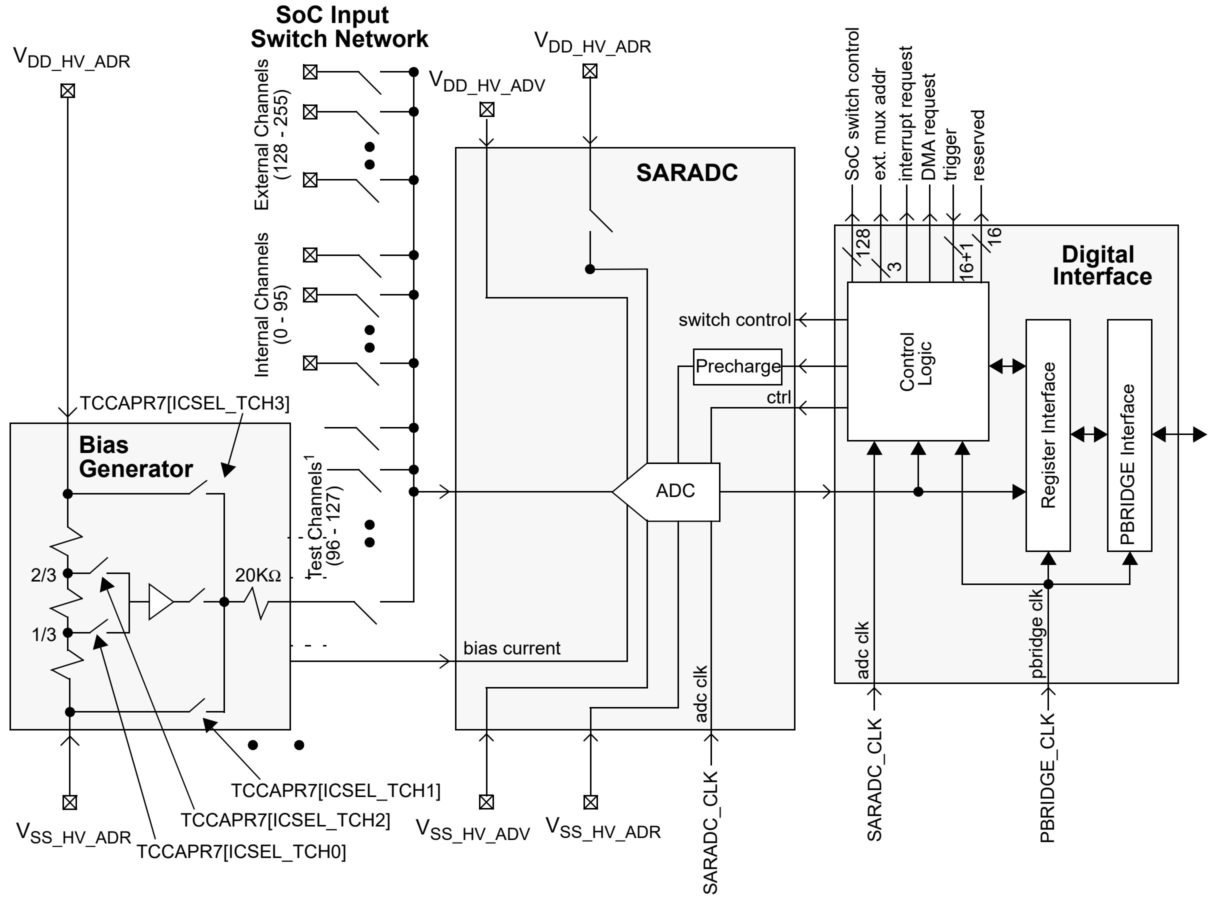
Note: The number of input channels and their mapping depend on the device configuration. Refer to the device-specific Analog-to-Digital Converters (ADC) Configuration section (analog input pin multiplexing and SAR ADC channel assignment) for a channel mapping table.

The mask registers present within the digital interface can be properly programed to configure which channel has to be converted. External channel selection is provided through external decode signals and are available as alternate functions on GPIO. Each set of eight external channels can be mapped to any internal analog channel via static software programming in specific registers.

Four conversion timing registers allow configuration of different precharging and sampling durations, one of which can then be assigned to each channel. Analog Watchdogs allow continuous hardware monitoring of analog input channels. The digital interface also provides for interrupt/DMA support for various conditions related to end of channel conversions.

Figure 390 shows the SARADC block diagram.

Figure 390. SARADC block diagram



Note1: Test channels are mapped to internal analog channels via the TCCAPRn registers of SAR ADC supervisor.

39.3 Feature description

Note: Not all features are implemented on all SARADC instances: refer to the ADC Configuration chapter for further information.

39.3.1 Main features

- Selectable 10-bit or 12-bit data resolution output for 12-bit SARADC
- Selectable 8-bit or 10-bit data resolution output for 10-bit SARADC
- Up to 96 internal channels, 32 test channels, 128 external channels supported; variable number of analog channels of each type controlled by parameters
- 4 different conversion timing registers selectable for any channel
- Mapping of external channel to any internal channel through static programming by software
- Shorting of test channel with internal channel through static programming by software
- External decode signals (3 signals) for selection of external analog mux inputs
- Normal conversion with One Shot/Scan modes
- Injected conversion with One Shot mode
- Normal conversion with dedicated trigger input
- Injected conversion with dedicated trigger input
- 2 different abort features that allow to abort either a single channel conversion or chain conversion
- Power Down Mode
- Dedicated data register for each channel, containing the following information:
  - Conversion result in one half-word:
10-bit or 12-bit for 12-bit SARADC
8-bit or 10-bit for 10-bit SARADC
  - Status byte which provides some conversion information such as mode of operation (Normal, Injected, or CTU), data valid, data overwritten status
  - Control byte for conversion timing parameter selection
- Configurable number of analog watchdogs.
- 2 different CTU modes (CTU Control mode and CTU “trigger mode”) available when CTU feature is present in SARADC instance.
- Fast Comparator Mode Conversion available when Fast Comparator feature is present in SARADC instance
- Smart Standby Wakeup Support available when Smart Standby Wakeup feature is present on SARADC instance
- Interrupt/DMA support for the following conditions
  - End of conversion of single channel for both normal, injected conversions
  - End of conversion chain for both normal, injected conversions
  - End of CTU conversion
  - Watchdog thresholds crossover

39.4 Functional description

Two main conversion types are available within the SARADC digital interface:

- Normal conversion
- Injected conversion

Note: "Precharge" option on this device means "de-charge" to reference ground VSS_HV_ADR_S.

39.4.1 Normal channel conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (ICNCMR0–2, TCNCMR, ECNCMR0–3). Each channel can be individually enabled by setting '1' in the corresponding field of these registers. Mask registers must be programed before starting the conversion and cannot be changed until the conversion of all the selected channels ends (MSR[NSTART] bit is reset).

39.4.1.1 Start of normal conversion

The normal conversion can be started as follows:

- By software
  - The normal conversion chain starts when the MCR[NSTART] bit is set. It is recommended that the normal trigger enable MCR[NTRGEN] bit is reset during conversions started by software. Once normal conversion has started, any further write operation of MCR[NSTART] bit has no effect during ongoing conversion, but the conversion is enqueued after completion of current conversion in one shot mode.
- By normal trigger
  - A normal trigger valid edge or level is detected to start the conversion.The normal trigger is enabled by setting the NTRGEN bit in MCR and two options are available. A programed event (rising, falling or both edges depending on NEDGESEL bit field of MCR) on the normal trigger input starts the normal conversion. Once normal conversion has started, any further valid trigger edge will be ignored during ongoing conversion. If the trigger mode is enabled (MCR[NTRGEN] bit is set) and a conversion is in progress, no conversion request should be given by programming MCR[NSTART] bit.
  - An appropriate gap should be maintained between any two consecutive triggers such that next trigger is received only after the completion of the current conversion. This gap value can be calculated as a sum of total conversion time as programed using CTR0-3 registers for each channel in the chain.

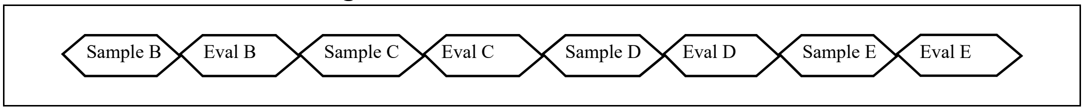
39.4.1.2 Normal conversion operating modes

Two operating modes are available for the normal conversion:

- One Shot Mode
- Scan Mode

To enter one of these modes, it is necessary to program the MCR[MODE] bit. The first phase of the conversion process involves sampling the analog channel and the next phase involves the conversion phase when the sampled analog value is converted to digital as shown in *Figure 391*.

Figure 391. Normal conversion flow



In **One Shot Mode** (MODE = 0), a sequential conversion specified in the ICNCMR0–2, TCNCMR, ECNCMR0–3 mask registers is performed only once. At the end of each conversion, the digital result of the conversion is stored into the corresponding data register.

For Example, channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the One Shot Mode. MODE = 0 is set for One Shot mode. Conversion starts from the channel B followed by conversion of channels D-E. At the end of conversion of channel E, the scanning of channels stops.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time MCR[NSTART] bit is reset if the conversion is started by software.

Once the normal conversion operation starts in one shot mode and another software conversion request is made writing the MCR[NSTART] bit and before MSR[NSTART] is cleared, one additional conversion is initiated with the same channel mask register settings after the current conversion is finished. This is called **enqueuing**. Enqueuing can help in removing the delay between two consecutive one shot chains and uncertainty caused by metastability between MCR[NSTART] bit written at system clock and generation of start pulse on ADC clock before every chain execution.

In **Scan Mode** (MODE = 1), a sequential conversion of N channels specified in the ICNCMR0–2, TCNCMR, ECNCMR0–3 registers is continuously performed. As in the previous case, at the end of each conversion the digital result of the conversion is stored into the corresponding data register.

In this mode, the software can start conversion by writing ‘1’ to MCR[NSTART] bit or a valid normal trigger edge is detected which also sets MCR[NSTART] bit. The MSR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One Shot Mode, the MCR[NSTART] bit is not reset. It can be reset by software when the user needs to stop scan mode. In this case, the current scan conversion is completed and MSR[NSTART] bit is also reset after the last conversion of the chain.

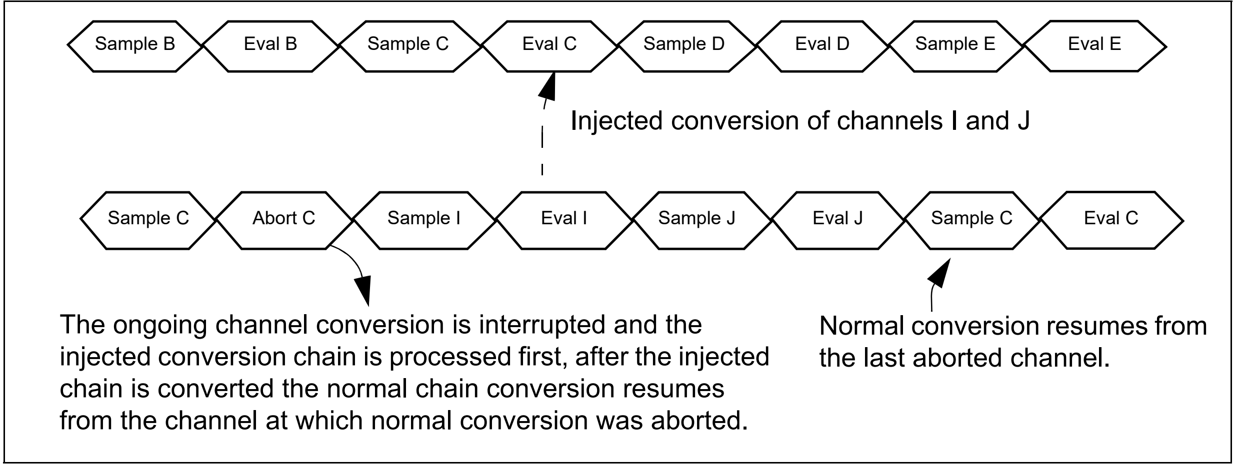
For Example, channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *Scan Mode*. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B starts followed by conversion of the channels D-E. This sequence repeats itself until the MCR[NSTART] bit is reset by software.

In both modes, at the end of each channel conversion an End Of Conversion interrupt is issued (if enabled by the corresponding mask bit), and at the end of the conversion sequence an End Of Chain interrupt is issued (if enabled by the corresponding mask bit). The End Of Chain interrupt is issued for each pass of scan mode.

39.4.2 Injected channel conversion

A conversion chain can be injected into the current normal conversion by configuring the injected mask registers (ICJCMR0–2, TCJCMR, ECJCMR0–3). As in normal conversion, each internal, test or external channel can be individually selected. This injected conversion can only be in one shot mode and interrupts the normal conversion. When an injected conversion is inserted, ongoing channel conversion is aborted and the injected channel request is processed. After the last channel in the injected chain is converted, normal conversion resumes from the channel at which the normal conversion was stopped, as shown in *Figure 392*.

Figure 392. Injected sample/conversion sequence



The injected conversion can be started as follows:

- By software
  - The injected conversion chain starts when the MCR[JSTART] bit is set. The current conversion is suspended and the injected chain is converted. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed. It is recommended that the injection trigger enable MCR[JTRGEN] bit is reset during conversions started by software. Once injected conversion has started, any further write operation of MCR[JSTART] bit will not have any effect during ongoing conversion and the new request will get enqueued and executed after current injected chain finishes.
- By injection trigger
  - The injection trigger is enabled by setting the JTRGEN bit in MCR. A programed event (rising, falling or both edges depending on JEDGESEL bit field of MCR) on the injection external trigger starts the injected conversion and also sets the MSR[JSTART] bit. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed.
  - When JTRGSEQ bit is set, the injection trigger sequence feature is enabled. In this mode, each valid edge of injection trigger input will convert one analog channel. First valid edge will convert the first analog channel enabled through ICJCMR0–2, TCJCMR, ECJCMR0–3 registers. The next valid edge converts the next enabled analog input and so on. Once all the enabled channels are converted, the channel selection loops back to the first enabled channel for next trigger edge. If the JTRGSEQ is reset, the channel selection is reset again to first channel enabled through ICJCMR0–2, TCJCMR, ECJCMR0–3 registers. If JTRGSEQ is reset before completing the injected chain, the remaining channels are completed in one-shot after receiving next valid trigger edge.
  - An appropriate gap should be maintained between any two consecutive injected triggers such that next injected trigger is received only after the completion of ongoing injected conversion. This gap value can calculated as a sum of total conversion time as programed using CTR0-3 registers for each channel in the injected chain.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time MSR[JSTART] is reset if the conversion is started by software.

At the end of each injected conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit) and at the end of the sequence an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit).

39.4.3 Abort conversion

Two different abort functions are provided:

- The user can abort the ongoing conversion by setting the ABORT bit in the MCR. The current conversion is aborted and the conversion of the next channel of the chain is immediately started (generating a new start pulse to the SARADC). In the case of an abort operation, the NSTART/JSTART bit remains set and the ABORT bit is reset after the conversion of the next channel starts. The EOC corresponding to the aborted channel is not generated. This behavior is true for normal or triggered/Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported generating an ECH interrupt. ABORT function should not be used when only single channel is selected in a conversion chain, instead ABORTCHAIN can be used.
- It is also possible to abort the current chain conversion by setting the ABORTCHAIN bit in the MCR. In that case the behavior of the ADC depends on the MODE bit. In fact, if scan mode is disabled, the NSTART bit is automatically reset together with the ABORTCHAIN bit. Otherwise, if the MODE bit is set to ‘1’, a new chain conversion is started. The EOC of the current aborted conversion is not generated but an ECH interrupt is generated to signal the end of the chain.

  When a chain conversion abort is requested (ABORTCHAIN bit is set) while an injected conversion is running over a suspended Normal conversion, there are 2 cases. If One Shot Mode is enabled (MODE=0 in MCR register) then both injected chain and Normal conversion chain are aborted (both the NSTART and JSTART bits are also reset). If the Scan Mode is enabled (MODE=1 in MCR register) then injected conversion is stopped and the suspended Normal conversion is restarting. (only JSTART bit is reset).

Note: ABORT/ABORTCHAIN functions are not supported during CTU initiated conversions.

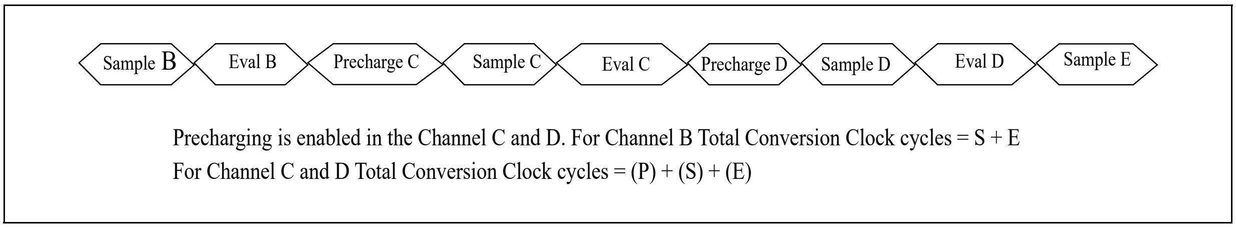
39.4.4 Analog conversion timings

39.4.4.1 Conversion timings

In order to support different loadings and switching times (in particular for the external channel types) four different Conversion Timing registers are present (CTR0–3). Each conversion timing register contains PRECHG and INPSAMP bit fields to program the required duration for precharging and sampling phases. The selection of these registers for each channel is done by the CTSEL bit field of corresponding channel data register.

Precharging phase allows to precharge or discharge the internal capacitor to a fixed internal voltage for duration determined by PRECHG bit field prior to actual analog channel sampling start. This is useful for resetting information regarding the last converted data. Precharging feature can be enabled or disabled for each channel by programming the PCE bit of corresponding channel data register. If precharging is enabled for a given channel, the normal sequence of operation is precharging, then sampling, then evaluation, as shown in *Figure 393*. Refer to device datasheet for the minimum precharge and sample durations required.

Figure 393. Analog conversion sequence



Bit fields PRECHG and INPSAMP are used to define the total conversion duration (t_{conv}) and in particular the partition among precharge duration (t_{prechg}), sampling phase duration (t_{sample}) and evaluation phase duration (t_{eval}).

The precharging phase duration is given by

$$t_{prechg} = PRECHG * t_{ck}$$

where PRECHG is the configured precharging phase duration and PCE bit of channel data register is ‘1’. If the PCE bit is ‘0’, the precharging phase is skipped and conversion starts with sampling phase directly.

The sampling phase duration is given by the following equation:

$$t_{sample} = INPSAMP * t_{ck}$$

where INPSAMP must be greater than or equal to 5 (hardware requirement). In case the value of INPSAMP is found to be less than 5, it is automatically set to 5 inside SARADC. The total evaluation phase duration is given by the following equation:

$$t_{eval} = EVAL * t_{ck}$$

In this phase, the internal sampling capacitor is disconnected from the analog channel and ADC estimates the digitized value of the sampled input using successive approximation algorithm. In the evaluation phase, all the bits are estimated sequentially to provide the conversion result.

The total conversion duration is (not including external multiplexing) is given by the following:

$$t_{conv} = t_{prechg} + t_{sample} + t_{eval}$$

The timings refer to the unit t_{ck} which is the inverse of f_{ck}, where f_{ck} = SARADC peripheral clock. Refer to the device data sheet for the total conversion time.

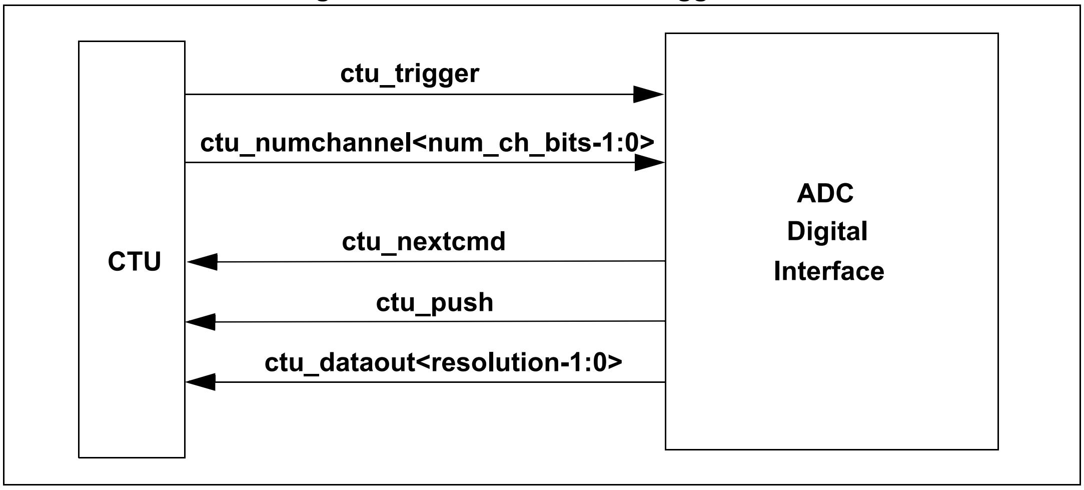
Note: The current implementation of ADCDIG does not support hardware restriction over values of PRECHG and INPSAMP fields. As minimum values for these fields are product specific and need to be managed through software.

39.4.5 Cross Triggering Unit (CTU) interface

The Cross Triggering Unit (CTU) is added to enhance the injected conversion capability of the SARADC. The CTU contains multiple event inputs that can be used to select the channels to be converted from the appropriate event configuration register. In Figure 394 the CTU/ADC interface is shown.The CTU generates a trigger output pulse of one clock cycle and outputs onto a data bus which channel has to be converted. A single channel is converted for each request. After performing the conversion, the digital interface returns the conversion result to CTU. The conversion result is also saved in the corresponding channel data register and it is compared with watchdog thresholds if requested. Together with the converted data, it also sends a request for next command and a control which can be used to push the conversion result into CTU data storage registers/FIFO.

These two pulses named *ctu_nextcmd* and *ctu_push* spaced of one or two clock cycle (of analog ADC clock). The ctu_nextcmd is set to ‘1’ during the last clock cycle or last but one of ADC evaluation phase (depending on ADC type used). The ctu_push is asserted when the conversion is finished meaning that the ctu_dataout is valid.

Figure 394. SARADC Cross Trigger Unit



Different CTU modes can be selected programming the CTUEN and CTU_MODE fields of the MCR with respect of CTSEN SARADC parameter as described in the following table.

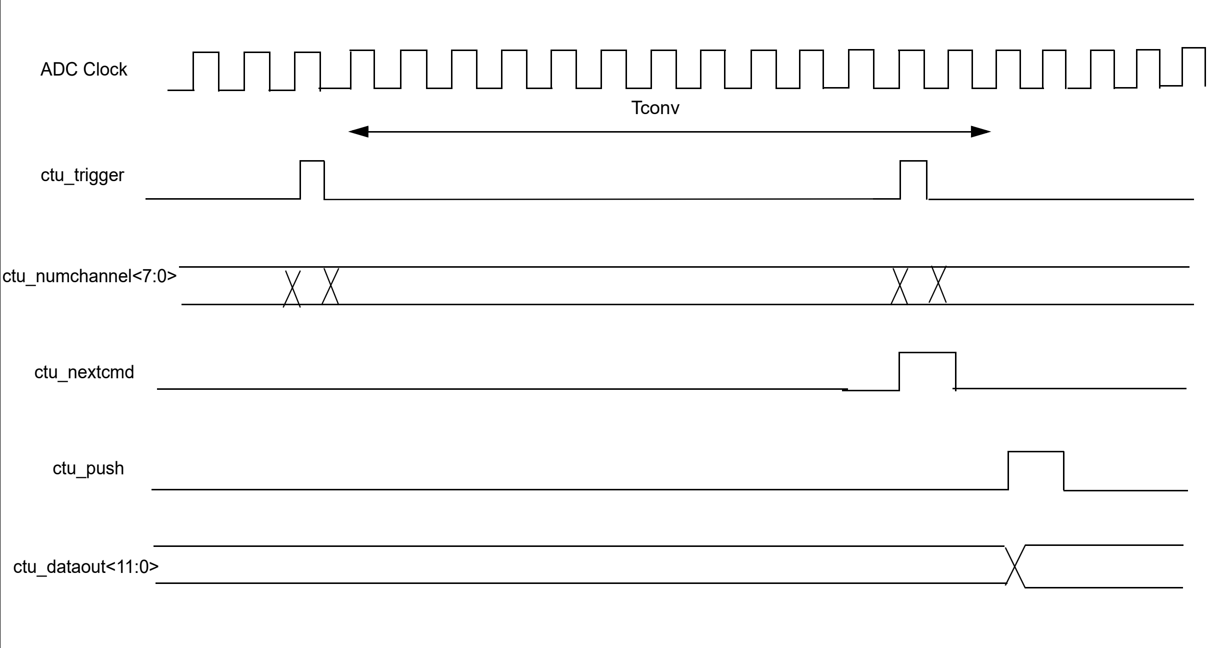
Table 465. CTU modes

CTSEN(1)	CTUEN	CTU_MODE	CTU mode
—	0	—	CTU is disabled
0	1	—	CTU trigger mode is enabled
1	1	—	CTU control mode is enabled
2	1	1	CTU trigger mode is selected
2	1	0	CTU control mode is selected

1. Refer to SAR ADC triggering section of chapter Analog-to-Digital Converters (ADC) Configuration for CTSEN value applicable on each SAR_ADC instance. CTSEN is hardware implemented, no bit register available.

The CTU, SARADC analog block and digital Interface are synchronous to each other, the CTU clock and ADC clock post division should be synchronous and phase aligned to ensure jitter free start pulses to ADC analog. Timings of CTU Interface signals are described in *Figure 395*.

Figure 395. CTU interface timings



39.4.5.1 CTU trigger mode

In CTU trigger mode, normal and injected conversions triggered by the CPU are still enabled. When a trigger pulse and the injected channel are received internally by the ADCDig, the conversion starts. The CTUSTART bit in the MSR is set automatically at this point and it is also automatically reset when the triggered injected conversion is completed.

In CTU trigger mode if an injected conversion (programed by the user by setting the JSTART bit) is ongoing and a pulse is received on the CTU trigger input line, then the injected channel conversion chain is aborted and only the triggered CTU injected conversion proceeds. Aborting the injected conversion, the JSTART bit in the MSR is reset to ‘0’. The JECH and JEOC are also reset to '0'. That abort is signaled through the status bit JABORTCHAIN in the MSR.

IN CTU trigger mode if normal conversion is ongoing and a pulse is received on the CTU trigger input line, then ongoing channel conversion is aborted and the triggered injected conversion is processed. When it is finished, normal conversion resumes from the channel at which the normal conversion was aborted.

If another CTU trigger pulse is received before the end of conversion, that request is discarded and this error condition is reported by CTUTRGERR bit in ISR.

When a normal conversion is requested during CTU conversion (CTUSTART bit = ‘1’), the normal conversion starts when CTU conversion is completed (CTUSTART reset). Otherwise, when an Injected conversion is requested during CTU conversion, that conversion is discarded and the JSTART bit in the MCR is immediately reset.

CTU trigger mode can be entered by setting CTUMODE=’1’ during power-down and CTUEN=’1’ any time after the power-down is exited. CTU trigger mode can be exited by clearing the CTUEN bit, But care must be taken not to exit the CTU mode when some CTU initiated conversion is ongoing. Also power down request (by setting MCR[PWDN] bit) should not be made while CTU transactions are ongoing.

39.4.5.2 CTU control mode

When CTU control mode is enabled, the CPU is able to write in the ADC registers but it cannot start any conversion. Conversion requests can be generated only by the CTU trigger pulse. If a normal or injected conversion is requested, it is automatically discarded.

When a CTU sends trigger pulse and channel number to be converted to SARADC digital interface, triggered injected conversion starts. The MSR[CTUSTART] bit is set automatically on the first received trigger pulse and it can be reset only when CTU is disabled (CTUEN bit to ‘0’).

CTU control mode can be entered by setting CTUMODE=’0’ during power-down and CTUEN=’1’ any time after the power-down is exited. CTU control mode can be exited by clearing the CTUEN bit, But care should be taken not to enter or exit the CTU mode when some conversion is ongoing. Also power down request (by setting MCR[PWDN] bit) should not be made while CTU transactions are ongoing.

39.4.6 Test channel connection with internal analog channel

The test channels are meant for monitoring various on-chip analog signals coming from temperature sensor, bandgap, voltage regulator and different voltage levels available on the chip. These analog signals can be interfaced to SARADC SUPERVISOR through test channels for testing purpose. Test channels are mapped from 96 to 127. Each test channel is controlled in the same way as normal internal channel.

FAST SAR ADC self-testing method: the test channel[127:124] is also mapped to various V_ref voltages (4 voltages) selection switches in the ADCBIAS block. The decoding of these channels to switch on and select a particular V_ref voltage is done in SoC level muxing logic. Refer ADC configuration for detailed mapping/control description of Self Test feature.(SAR_ADC_10bit_STDBY has no test channels)

It is also possible to perform test channel conversions with and without shorting the normal analog channel pin with the internal test voltage.

For analog test channel assignment, refer to the device configuration or ADC configuration section.

After the conversion is performed on a test channel the converted data is stored at two CDR locations at the end of conversion, first location is the CDR belonging to the test channel on which conversion is performed and second location is the CDR belonging to internal channel on which test channel is mapped.

39.4.7 External channel mapping to internal analog channel

External channels are supposed to be used in conjunction with a given internal analog channel along with an 8:1 external mux outside the device. Each input of the external mux can be mapped to a unique channel number, thus allowing it to perform conversions like any other channel of the ADC. External channels are mapped from 128 to 255.

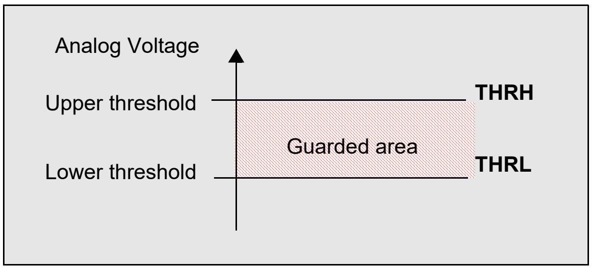
External channels are implemented in the multiples of 8. Each slice of eight channels can be associated to any physical internal analog channel mapped from 0 to 95. This external-to-internal channel mapping is determined by the ICSEL_ECx_x bit field of ECMICR0–3 registers. For example, when external multiplexer is planned to be used on internal channel 5, and the external inputs need to be mapped to channels 128 to 135, then this channel number 5 must be written to the ICSEL_ECH128_135 bit field of the ECMICR0 register. After reset, all external channels are mapped to internal analog channel 0. If a conversion on a single external channel (out of group of 8) mapped to a particular internal channel has to be performed than only that particular channel should be enabled in the Conversion Mask Register (ECNCMR or ECJCMR).

After the conversion is performed on a external channel, the converted data is stored at two CDR locations at the end of conversion. The first location is the CDR belonging to the external channel on which conversion is performed and the second location is the CDR belonging to the internal channel on which the external channel is mapped.

39.4.8 Programmable analog watchdog

Analog watchdogs are used for determining whether the result of the conversion of a channel lies within a given guard area (as shown in the *Figure 396*) bounded by upper and a lower threshold values named THRH and THRL respectively.

Figure 396. Analog watchdog guarded area



Following conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guard area, then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the WTISR as explained in *Table 466*. Depending on the MSKWDGxL and MSKWDGxH mask bits in the WTIMR, an interrupt is generated on threshold violation.

Table 466. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRH ≤ converted data ≤ THRL

The lower and higher threshold values for the analog watchdog are programed using THRHLR0–15 registers. Analog watchdog functionality for each channel can be enabled independently from ICWENR0–2, TCWENR, ECWEN0–3 registers and can select the watchdog threshold registers (THRHLR0–15) to be used by programming ICWSELR0–11, TCWSELR0–3, ECWSELR0–15 registers. The threshold registers selected by WSEL_CHx field of watchdog threshold selection registers will provide the threshold values.

For example, if channel 30 is to be monitored with the threshold values in register THRHLR12, then WSEL_CH30 is programed to select THRHLR12 register to provide threshold values. Enabling is done by setting the bit corresponding to channel 30 in ICWENR0 register.

If the converted value for a particular channel lies outside the range specified by threshold values, then the corresponding bit is set in ICAWORR0–2, TCAWORR, ECAWORR0–3 registers. In this case, a set of threshold registers (THRHLR0–15) can be linked to several analog channels. The threshold values to be selected for a channel needs be programed only once in ICWSELR, TCWSELR, ECWSELR registers.

Note: If the higher threshold for the analog watchdog is programed lower than the lower threshold and the converted value is less than the lower threshold, then the WDGxL interrupt for the low threshold violation is set, else if the converted value is greater than the lower threshold (consequently also greater than the higher threshold) then the interrupt WDGxH for high threshold violation is set. The user should therefore take care to avoid this situation as it could lead to misinterpretation of the watchdog interrupts.

39.4.9 DMA functionality

A Direct Memory Access (DMA) request can be programed after the conversion of every channel by setting the respective masking bit in ICDSR0–2, TCDSR and ECDSR0–3 registers. The DMA masking registers must be programed before starting any conversion. The DMA transfers can be enabled using the DMAE[DMAEN] bit. When the DMAE[DCLR] bit is set, the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

39.4.10 Interrupts

The SARADC digital interface generates the following maskable interrupt signals:

- End Of Conversion interrupts
  - NEOC (end of normal conversion of each channel)
  - NECH (end of normal chain)
  - JEOC (end of injected conversion of each channel)
  - JECH (end of injected chain)
  - EOCTU (End of CTU Conversion)

The EOC_CH[x] bit of interrupt pending registers (ICIPR0–2, TCIPR, ECIPR0–3) is set when channel x completes the conversion (normal or injection). This pending status is qualified when the corresponding mask bit IM_CH[x] is also set in the interrupt mask registers (ICIMR0–2, TCIMR, ECIMR0–3).

The NEOC interrupt is generated only if the following conditions are met

- ISR[NEOC] bit is set after the normal conversion completion for any channel x.
- IMR[MSKNEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

The JEOC interrupt is generated only if the following conditions are met

- ISR[JEOC] bit is set after the injected conversion completion for any channel x.
- IMR[MSKJEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

The EOCTU interrupt is generated only if the following conditions are met

- ISR[EOCTU] bit is set after the CTU triggered conversion completion for any channel x.
- IMR[MSKEOCTU] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

It is recommended to clear the NEOC/JEOC/EOCTU bits of ISR and all the individual channel pending bits which are not masked while servicing the interrupt.

The ISR[NECH] bit is set when a normal channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

The ISR[JECH] bit is set when an injected channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

- WDGxL and WDGxH (Watchdog threshold) interrupt requests

The interrupts generated due to the analog watchdog are handled by registers WTISR and WTIMR in order to check and enable the interrupt requests to external interrupt controller module. The watchdog interrupt source sets two pending bits WDGxH and WDGxL for each channel being monitored in the WTISR. The interrupt request is generated if at least one of the following conditions are met:

- WTISR[WDGxL] bit is set and the corresponding mask bit MSKWDGxL is also set in WTIMR for at least one channel.
- WTISR[WDGxH] bit is set and the corresponding mask bit MSKWDGxH is also set in WTIMR for at least one channel.

The ISR and WTISR contain the interrupt pending request status flags. In case the user wants to clear a particular interrupt event status, writing a ‘1’ to the corresponding status bit clears the pending interrupt flag. (At this write operation, all other bits of the ISR must be maintained at ‘0’.)

The ADC module provides dedicated interrupt outputs for all interrupt sources. Also, all interrupt lines are OR-ed together to provide a single output to the external interrupt controller if a combined interrupt is required.

Conversion should not be initiated unless the required conversion mask bits (ECNCMR or ECJCMR) are set.

39.4.11 External decode signals selection and delay

The ADC provides three external decode signals used as select lines for 8:1 external mux. For each set of 8 external channels (for example [CH128:CH135], [CH136:CH143], and so on), the same 3-bit decode signals output are used to select the inputs of external mux. These external mux select lines change in grey code sequence for each increment of channel selection. This ensures that only one line changes and avoids any current spikes or leakage if more than one line changes simultaneously.

In order to take into account the control switching time of the external analog mux, a Decode Signals Delay register is provided. Writing that register allows programming of the delay between the decoding signal selection and the actual start of conversion.

39.4.12 Power down mode

The SARADC analog block can be put in low current consumption mode by setting the PWDN bit in MCR. This is the default mode after exiting from reset. This state must be exited before starting any operation by resetting the PWDN bit in the MCR. When in power-down mode, no conversion should be started. Bit ADCSTATUS[0] in the MSR is set only when ADC enters power-down mode.

The power-down mode can be requested by setting the PWDN bit in the MCR. If a one-shot conversion is ongoing, the SARADC cannot immediately enter the power-down mode. In fact, the SARADC enters power-down mode only after completing the ongoing conversion. If a scan conversion is ongoing, the ongoing operation shall be first aborted manually by resetting the NSTART bit and using the ABORTCHAIN bit. A power Down request while scan mode conversion is ongoing is ignored.

After the power down phase is completed, the process ongoing before the power down phase must be restarted manually (by setting the appropriate START bit). Resetting PWDN bit and setting NSTART or JSTART bit during the same cycle is forbidden.

CTU triggered conversion should not be initiated during power down mode. In both CTU modes, to request power down mode the application has to wait for the end of the current CTU conversion (CTUSTART bit automatically reset when conversion finishes). Additionally, when SARADC is in CTU control mode the application needs to exit the CTU control mode (by resetting the CTUEN bit) before requesting the power down mode.

39.4.13 Stop mode

The SARADC can be requested to enter this low power mode anytime during the operation. If stop mode is requested while some Normal/Injected mode chained conversions are ongoing, the current channel conversion will be completed and upon stop mode exit the conversion will be resumed from next channel. However, stop mode should not be requested while CTU triggered conversions are ongoing.

39.4.14 Fast Comparator mode conversion

In this mode the SARADC analog block/sequencer can be requested to work as a fast comparator instead of a regular data converter for the initiated conversion (single/chain). Here, the sample phase remains unchanged, but the evaluation phase is reduced to just two cycles of fast comparison of the sampled channel value with regards to the corresponding threshold pair values (THRL,THRH) provided at the start of each conversion.

39.4.14.1 Execution order

The programming sequence and execution order when SARADC works in Fast Comparator mode is as follows:

1. The user must configure and program the required SARADC configuration and control registers like MCR, Channel Conversion Mask (CMRs), CTRs in the same way and order as if normal or injected conversion were initiated. CDR/CTR programming constraint is the following:
   CTR[INPSAMP] >= 2. Value of 2 for the minimum sampling duration.
2. Enable the Fast Comparator feature by programming the Fast Comparator Enable (MCR[FCE]) bit.
3. Associate and select (program WSELR) 1 out of maximum 16 watchdog threshold pairs (program WTHRHLR[THRL,THRH]) for each of the channels selected for fast

comparison and enable (WENR). WTHTHLR[THRL,THRH] programming constraints are as follows:

a) THRH > THRL

b) THRH < Maximum value of converted analog signal, that is $2^{resolution}-1$

c) THRL > Minimum value of converted analog signal, that is 0

4. Initiate the conversion by setting MCR[NSTART/JSTART].
5. The result of fast comparison from ADC analog block {THRH+1, THRL-1 or (THRH+THRL)/2} and the comparator error status are stored as normal conversion results in the CDR registers.
6. The conversion results are again compared by the watchdog monitor inside the ADC digital interface and resulting violation is reported back to the system by WTISR[WDG0-15L/H] status bits, corresponding interrupts and two external signals (upper threshold violation, lower threshold violation).
7. For additional requirements about smart standby wake up using fast comparator mode, refer to the below *Section 39.4.15* for more details.

39.4.15 Smart standby wake up support

Here, the SARADC in conjunction with the CTU and the smart standby wake up unit (SSWU) can be used to wake up the system by regularly checking the sampled values. This is done with respect to the programed thresholds on certain selected channels.

39.4.15.1 Execution order

The programming sequence and execution order when SARADC works in conjunction with the SSWU is as follows:

1. User must configure and program the required SARADC configuration and control registers like MCR, Channel Conversion Mask (CMRs), CTRs.
2. Associate and select (program WSELR) 1 out of maximum 16 watchdog threshold pairs (program WTHRHLR[THRL,THRH]) for each of the channels selected through CMRs and enable (WENR).
3. Set MCR[FCE]; if fast comparator mode conversion is required (optional), the steps 1, 2 and 3 should be performed before the system enters in standby mode.
4. The CTU and SSWU exit the power down mode of SARADC by de-asserting the power down request signal.
5. CTU Initiates conversion providing CTU triggers. The minimum gap between the power down request signal pulse and the first CTU trigger pulse should be greater than 2 to 3 ADC clock cycles.
6. The result of conversion is available in CDR after the end of conversion pulse arrives from SARADC analog block; If MCR[FCE] is set, comparison results are stored as either THRH+1, THRL-1 or (THRH+THRL)/2 in the CDR registers.
7. The conversion results are again compared by the watchdog monitor inside the ADC digital interface and the resulting violation is reported back to the SSWU by two external signals (upper threshold violation, lower threshold violation).
8. SSWU can request for SARADC power-down mode entry by asserting the power down request signal (rising edge) port.

39.5 Memory map and registers

This section provides memory map for the SARADC digital interface.

Note: The actual availability and number of registers and their configuration are chip-specific. For this information, refer to "ADC Configuration" chapter.

Table 467. SARADC digital interface register map

Offset	Register	Location
0x000	Main Configuration Register (MCR)	Section 39.5.1.1
0x004	Main Status Register (MSR)	Section 39.5.1.2
0x008–0x00C	Reserved	
0x010	Interrupt Status Register (ISR)	Section 39.5.1.3
0x014–0x01C	Internal channel Interrupt Pending Registers 0–2 (ICIPR0–ICIPR2)	Section 39.5.1.4
0x020	Interrupt Mask Register (IMR)	Section 39.5.1.5
0x024–0x02C	Internal Channel Interrupt Mask Registers 0–2 (ICIMR0–ICIMR2)	Section 39.5.1.6
0x030	Watchdog Threshold Interrupt Status Register (WTISR)	Section 39.5.1.7
0x034	Watchdog Threshold Interrupt Mask Register (WTIMR)	Section 39.5.1.8
0x038–0x03F	Reserved	
0x040	DMA Enable Register (DMAE)	Section 39.5.1.9
0x044–0x04C	Internal Channel DMA Select Registers 0–2 (ICDSR0–ICDSR2)	Section 39.5.1.10
0x050–0x05C	Reserved	
0x060–0x06C	Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)	Section 39.5.1.11
0x070–0x090	Reserved	
0x094–0x0A0	Conversion Timing Registers 0–3 (CTR0–3)	Section 39.5.1.12
0x0A4–0x0AC	Internal Channel Normal Conversion Mask Registers 0–2 (ICNCMR0–ICNCMR2)	Section 39.5.1.13
0x0B0	Reserved	
0x0B4–0x0BC	Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–ICJCMR2)	Section 39.5.1.14
0x0C0–0x0C4	Reserved	
0x0C8	Power Down Exit Delay Register (PDEDR)	Section 39.5.1.15
0x0CC–0x0FC	Reserved	
0x100–0x27C	Internal Channel Data Registers 0–95 (ICDR0–ICDR95)	Section 39.5.1.16
0x280–0x2AC	Watchdog Threshold Registers 4-15 (WTHRHLR4-WTHRHLR15)	Section 39.5.1.17
0x2B0–0x2DC	Internal Channel Watchdog Selection Registers 0–11 (ICWSELR0–ICWSELR11)	Section 39.5.1.18
0x2E0–0x2E8	Internal Channel Watchdog Enable Registers 0–2 (ICWENR0–ICWENR2)	Section 39.5.1.19
0x2EC–0x2EF	Reserved	

Table 467. SARADC digital interface register map (continued)

Offset	Register	Location
0x2F0–0x2F8	Internal Channel Analog Watchdog Out of Range Registers 0–2 (ICAWORR0–ICAWORR2)	Section 39.5.1.20
0x2FC–0x3FC	Reserved	
0x400	Test Channel Interrupt Pending Register (TCIPR)	Section 39.5.1.21
0x404	Test Channel Interrupt Mask Register (TCIMR)	Section 39.5.1.22
0x408	Test Channel DMA Select Register (TCDSR)	Section 39.5.1.23
0x40C	Test Channel Normal Conversion Mask Register (TCNCMR)	Section 39.5.1.24
0x410	Test Channel Injected Conversion Mask Register (TCJCMR)	Section 39.5.1.25
0x414–0x420	Test Channel Watchdog Selection Registers 0–3 (TCWSELR0–TCWSELR3)	Section 39.5.1.26
0x424	Test Channel Watchdog Enable Register (TCWENR)	Section 39.5.1.27
0x428	Test Channel Analog Watchdog Out of Range Register (TCAWORR)	Section 39.5.1.28
0x42C	Reserved	
0x430–0x44C	Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)	Section 39.5.1.29
0x450–0x4CC	Test Channel Data Registers 96-127 (TCDR96-TCDR127)	Section 39.5.1.30
0x4D0–0x4FC	Reserved	
0x500	External Channel Decode Signals Delay Register (ECDSDR)	Section 39.5.1.31
0x504–0x50C	Reserved	
0x510–0x51C	External Channel Interrupt Pending Registers 0–3 (ECIPR0–ECIPR3)	Section 39.5.1.32
0x520–0x52C	External Channel Interrupt Mask Registers 0–3 (ECIMR0–ECIMR3)	Section 39.5.1.33
0x530–0x53C	External Channel DMA Select Registers 0–3 (ECDSR0–ECDSR3)	Section 39.5.1.34
0x540–0x54C	External Channel Normal Conversion Mask Registers 0–3 (ECNCMR0–ECNCMR3)	Section 39.5.1.35
0x550–0x55C	External Channel Injected Conversion Mask Registers 0–3 (ECJCMR0–ECJCMR3)	Section 39.5.1.36
0x560–0x59C	External Channel Watchdog Selection Registers 0–15 (ECWSELR0–ECWSELR15)	Section 39.5.1.37
0x5A0–0x5AC	External Channel Watchdog Enable Registers 0–3 (ECWENR0–ECWENR3)	Section 39.5.1.38
0x5B0–0x5BC	External Channel Analog Watchdog Out of Range Registers 0–3 (ECAWORR0–ECAWORR3)	Section 39.5.1.39
0x5C0–0x5CC	External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–ECMICR3)	Section 39.5.1.40
0x5D0–0x7CC	External Channel Data Registers 128-255 (ECDR128-ECDR255)	Section 39.5.1.41

39.5.1 Register descriptions

39.5.1.1 Main Configuration Register (MCR)

Offset: 0x000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OWREN	WLSIDE	MODE	Reserved	NSTART	NTRGEN[3]	NEDGESEL[3]		JSTART	JTRGEN[2]	JEDGESEL[2]		JTRGSEQ[2]	0	CTUEN[1]	CTU_MODE[1]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	0	0	0	JTRGSEL[2]				ABORTCHAIN	ABORT	0	FRZ	0	FCE[4]	EDCSELF[5]	PWDN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Only available with cross-triggering unit (CTU) feature.
2. Only available with injection trigger feature.
3. Only available with normal trigger feature.
4. Only available with Fast Comparator feature.
5. Only available with External decode format selectable feature.

Figure 397. Main Configuration Register (MCR)

Table 468. MCR field descriptions

Field	Description
0 OWREN	Overwrite enable This bit enables or disables the functionality to overwrite unread converted data in ICDR, TCDR, ECDR registers. 0 Prevents overwrite of unread converted data, new result is discarded. 1 Enables converted data to be overwritten by a new conversion.
1 WLSIDE	Write left/right-aligned 0 The conversion data in ICDR, TCDR, ECDR registers is written right-aligned. 1 Conversion data is left-aligned (from 15 to (15 – resolution + 1))
2 MODE	One Shot/Scan 0 One Shot Mode: Configures the normal conversion of one chain. 1 Scan Mode: Configures continuous chain conversion mode; when the programed chain conversion is finished it restarts immediately

Table 468. MCR field descriptions (continued)

Field	Description
4 NSTART	Normal Start conversion Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation. 0 Causes the current chain conversion to finish and stops the operation. 1 Starts the chain or scan conversion **Note:** The application relies on the corresponding bit in MSR (MSR[NSTART]) to confirm that any changes to this bit has taken effect. Example with NSTART bit: MCR[NSTART] = 1; // Register to start normal conversion while(MSR[NSTART]==0); // Ensure normal conversion is completed before raising new request
5 NTRGEN	Normal trigger enable 0 Normal trigger disabled to start a normal conversion. 1 Normal trigger enabled to start a normal conversion.
6:7 NEDGESEL	Normal trigger edge selection 00 Falling edge of normal trigger is selected. 01 Rising edge of normal trigger is selected. 10 Both rising and falling edges of normal trigger are selected. (*x* = 0,1). 11 Both rising and falling edges of normal trigger are selected. (*x* = 0,1)
8 JSTART	Injected Start conversion Setting this bit will start the configured injected analog channels to be converted by software. Resetting this bit has no effect, as the injected chain conversion cannot be interrupted. 0 Writing ‘0’ has no effect. 1 Starts the injected chain conversion. **Note:** The application should rely on the corresponding bit in MSR (MSR[JSTART]) to confirm that any changes to this bit have taken effect.
9 JTRGEN	Injection trigger enable 0 Injection trigger disabled for channel injection (injected conversion cannot be started using an injection trigger) 1 Injection trigger enabled for channel injection
10:11 JEDGESEL	Injection trigger edge selection 00 Falling edge of injection trigger is selected 01 Rising edge of injection trigger is selected 10 Both rising and falling edges of injection trigger are selected. (*x* = 0,1) 11 Both rising and falling edges of injection trigger are selected. (*x* = 0,1)
12 JTRGSEQ	Injection trigger sequence enable 0 Injection trigger sequence mode is disabled 1 Injection trigger sequence mode is enabled
14 CTUEN	Cross Trigger Unit Enable 0 The cross triggering unit is disabled and the triggered injected conversion cannot take place 1 The cross triggering unit is enabled and the triggered injected conversion can take place **Note:** Can be set or cleared anytime with some restriction mentioned in CTU trigger/control mode section.

Table 468. MCR field descriptions (continued)

Field	Description
15 CTU_MODE	Cross Trigger Unit Mode This bit is present only if generic parameter CTSEN = 2, otherwise this bit is a reserved bit and always read as 0. This should be set only when ADC Digital is in power-down mode. 0 CTU control mode is selected 1 CTU trigger mode is selected
20:23 JTRGSEL	Injection Trigger Input Selection This bit field selects which trigger input to be selected to start injected conversion. 0000 Trigger input 0 is selected. 0001 Trigger input 1 is selected ... 1111 Trigger input 15 is selected **Note:** JTRGSEL[3:0] must be 0000 when only one trigger input (input 0) is supported
24 ABORTCHAIN	Abort Chain When this bit is set, the ongoing Chain Conversion is aborted. This bit is reset by hardware as soon as a new conversion is requested. 0 Conversion is not affected 1 Aborts the ongoing chain conversion **Note:** Behavior of this request during CTU mode is mentioned in CTU trigger/control mode section.
25 ABORT	Abort Conversion When this bit is set, the ongoing conversion is aborted and a new conversion is invoked. This bit is reset by hardware as soon as a new conversion is invoked. 0 Conversion is not affected 1 Aborts the ongoing conversion **Note:** Behavior of this request during CTU mode is mentioned in CTU trigger/control mode section.
27 FRZ	Freeze This bit enables to stop the SARADC conversions at the end of current channel conversion when SoC enters debug mode. 0 Conversions are not stopped 1 When SoC enters debug mode, further conversions by SARADC analog block will be stopped. The ongoing channel conversion will be completed and the converted data is stored. When the SoC exits debug mode, SARADC resumes the job that was left before halting the conversions. Changing configuration registers during halted condition may cause a pending operation to fail when normal operation is resumed
29 FCE	Fast Comparator Enable This bit enables the fast comparison mode for upcoming channel conversions. 0 Fast Comparison mode disabled 1 Fast Comparison mode enabled

Table 468. MCR field descriptions (continued)

Field	Description
30 EDCSELF	External Decode Channel Format Select (EDCSELF) This bit selects the format for 3-bit output port ipp_decode_extch used as select line for external decode channel 8:1 muxes. The last three bits of external channel number are passed as is (if value 0) or converted to gray code (if value 1) and passed to this port. 0 Binary format 1 Gray Code format
31 PWDN	Power-down enable When this bit is set, the analog module is requested to enter Power Down mode. When SARADC status is PWDN, resetting this bit starts SARADC transition to IDLE mode. 0 SARADC is in normal mode. 1 SARADC has been requested to power down. **Note:** When requesting a power mode transition, the application should rely on the MSR[ADCSTATUS] bit field in order to confirm that the transition has taken effect.

39.5.1.2 Main Status Register (MSR)

Offset: 0x004 Access: User Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHADDR								0	0	0	0	0	ADCSTATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Only available with cross-triggering unit (CTU) feature.

Figure 398. Main Status Register (MSR)

Table 469. MSR field descriptions

Field	Description
4 NSTART	This status bit is used to signal that a normal conversion is ongoing. 0 Normal conversion is not taking place. 1 Normal conversion is ongoing or the normal conversion is pending due to injected conversion or CTU triggered injected conversion.
8 JSTART	This status bit is used to signal that an injected conversion is ongoing. 0 Injected conversion is not taking place. 1 Injected conversion is ongoing.
13 JABORTCHAIN	This status bit is used to signal that an Injected conversion chain has been aborted. This bit is reset when a new conversion starts. 0 Last injected conversion chain has not been aborted. 1 Last injected conversion chain has been aborted.
15 CTUSTART	This status bit is used to signal that a CTU conversion is ongoing. This bit is set when a CTU trigger pulse is received and the CTU conversion starts. When CTU trigger mode is enabled this bit is automatically reset when the conversion is completed. Otherwise, if Control Mode is enabled this bit can be reset only when the CTU is disabled (CTUEN set to ‘0’). 0 CTU triggered injected conversion is not taking place. 1 CTU triggered injected conversion is ongoing.
16:23 CHADDR	Channel under measure address This status bit is used to signal which channel is under measure.
29:31 ADCSTATUS	The value of this parameter depends on ADC status. While requesting an external channel conversion, SARADC digital interface needs to wait for a fixed time determined by DSD bit field of DSDR before proceeding to actual sampling phase. This is known as wait state. 000 IDLE 001 Power-down 010 Wait state 011 — 100 Sample 101 — 110 Conversion 111 —

39.5.1.3 Interrupt Status Register (ISR)

Offset: 0x010 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR(1)	EOCTU(1)	JEOC	JECH	NEOC	NECH
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only available with cross-triggering unit (CTU) feature.

Figure 399. Interrupt Status Register (ISR)

Table 470. ISR field descriptions

Field	Description
26 CTUTRGERR	CTU Trigger Error Flag This Error Flag indicates that the next CTU trigger has been discarded as it overlapped during execution of current CTU conversion. This indicated that CTU sampling duration is less than programed conversion time of ADC. 0 No CTU trigger overlap error condition since last CTU conversion. 1 CTU trigger overlap error has occurred.
27 EOCTU	End of CTU channel conversion interrupt flag This bit indicates the end of conversion for a CTU injected channel. 0 End of conversion of CTU triggered channel has not occurred since the last time the flag is cleared. 1 End of conversion of CTU triggered channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
28 JEOC	End of injected channel conversion interrupt flag This bit indicates the end of conversion for an injected channel. 0 End of conversion of injected channel has not occurred since the last time the flag is cleared. 1 End of conversion of injected channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
29 JECH	End of injected chain conversion interrupt flag This bit indicates the end of conversion for an injected chain. 0 End of conversion of injected chain has not occurred since the last time the flag is cleared. 1 End of conversion of injected chain has occurred or the flag has not been cleared since the last end of conversion occurrence.

Table 470. ISR field descriptions (continued)

Field	Description
30 NEOC	End of normal channel conversion interrupt flag This bit indicates the end of conversion for a normal channel. 0 End of conversion of normal channel has not occurred since the last time the flag is cleared. 1 End of conversion of normal channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
31 NECH	End of normal chain conversion interrupt flag This bit indicates the end of conversion for an normal chain. 0 End of conversion of normal chain has not occurred since the last time the flag is cleared. 1 End of conversion of normal chain has occurred or the flag has not been cleared since the last end of conversion occurrence.

39.5.1.4 Internal Channel Interrupt Pending Register *n* (ICIPR*n*)

The interrupt channel register to channel association is described in *Table 472*.

Offset: 0x014 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 400. Internal Channel Interrupt Pending Register *n* (ICIPR*n*)

Table 471. ICIPR*n* field descriptions

Field	Description
0:31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 472. ICIPR0–2 Registers to channel association

Register	Register bits 0:31
ICIPR0	EOC_CH[31:0]
ICIPR1	EOC_CH[63:32]
ICIPR2	EOC_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field EOC_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.5 Interrupt Mask Register (IMR)

The description of the Interrupt Mask Register is given below.

Offset: 0x020 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU(1)	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only available with cross-triggering unit (CTU) feature.

Figure 401. Interrupt Mask Register (IMR)

Table 473. IMR field descriptions

Field	Description
27 MSKEOCTU	Mask bit for EOCTU 0 EOCTU interrupt is disabled. 1 EOCTU interrupt is enabled.
28 MSKJEOC	Mask bit for JEOC 0 JEOC interrupt is disabled. 1 JEOC interrupt is enabled.
29 MSKJECH	Mask bit for JECH 0 JECH interrupt is disabled. 1 JECH interrupt is enabled.
30 MSKNEOC	Mask bit for NEOC 0 NEOC interrupt is disabled. 1 NEOC interrupt is enabled.
31 MSKNECH	Mask bit for NECH 0 NECH interrupt is disabled. 1 NECH interrupt is enabled.

39.5.1.6 Internal Channel Interrupt Mask Register *n* (ICIMR*n*)

The interrupt mask register to channel association is described in *Table 475*.

Offset: 0x024 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 402. Internal Channel Interrupt Mask Register *n* (ICIMR*n*)

Table 474. ICIMR*n* field descriptions

Field	Description
0:31 IM_CH[x]	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 475. Interrupt Mask Registers to Channel Association

Register	Register bits 0:31
ICIMR0	IM_CH[31:0]
ICIMR1	IM_CH[63:32]
ICIMR2	IM_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field IM_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.7 Watchdog Threshold Interrupt Status Register (WTISR)

This register gives the interrupt status information for the 16 possible upper and 16 possible lower threshold limits monitored by up to 16 watchdog monitors which can be selected for each channel.

Offset: 0x030 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 403. Watchdog Threshold Interrupt Status Register (WTISR)

Table 476. WTISR field descriptions

Field	Description
30-2x WDGxH [x = 0–15]	This corresponds to the interrupt generated on the converted value being higher than the programed higher threshold as reported by WDG monitor ‘x’. 0 Converted data is not higher than the programed higher threshold. 1 Converted data is higher than the programed higher threshold.
31-2x WDGxL [x = 0–15]	This corresponds to the interrupt generated on the converted value being lower than the programed lower threshold as reported by WDG monitor ‘x’. 0 Converted data is not lower than the programed lower threshold. 1 Converted data is lower than the programed lower threshold.

39.5.1.8 Watchdog Threshold Interrupt Mask Register (WTIMR)

This register gives the interrupt mask information for the 16 possible upper and 16 possible lower threshold limits which can be selected for each channel.

Offset: 0x034 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 404. Watchdog Threshold Interrupt Mask Register (WTIMR)

Table 477. WTIMR field descriptions

Field	Description
30-2x MSKWDGxH [x = 0–15]	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programed higher threshold. 0 Interrupt for WDGxH is disabled. 1 Interrupt for WDGxH is enabled.
31-2x MSKWDGxL [x = 0–15]	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programed lower threshold. 0 Interrupt for WDGxL is disabled. 1 Interrupt for WDGxL is enabled.

39.5.1.9 DMA Enable Register (DMAE)

The DMA Enable (DMAE) register sets up the DMA for use with the SARADC.

Offset: 0x040 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
W															DCLR	DMAEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 405. DMA Enable Register (DMAE)

Table 478. DMAE register field descriptions

Field	Description
30 DCLR	DMA clear sequence enable 0 DMA request cleared by Acknowledge from DMA controller. 1 DMA request cleared on read of data registers.
31 DMAEN	DMA global enable 0 DMA feature disabled. 1 DMA feature enabled.

39.5.1.10 Internal Channel DMA Select Register *n* (ICDSR*n*)

The DMA select registers to channel association is described in *Table 480*.

Offset: 0x044 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 406. Internal Channel DMA Select Register *n* (ICDSR*n*)

Table 479. ICDSR*n* field descriptions

Field	Description
0:31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 480. Internal Channel DMA Select Registers to Channel Association

Register	Register bits 0:31
ICDSR0	DS_CH[31:0]
ICDSR1	DS_CH[63:32]
ICDSR2	DS_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field DS_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.11 Watchdog Threshold Register *n* (WTHRHLR*n*)

Note: The number of WTHRHLR registers corresponds to the number of analogs watchdogs implemented in the ADC module.

The length of the threshold field depends on the SARADC resolution. For 10-bit SARADC, the bits 4, 5, 20 and 21 are reserved.

Note: The reset value of the bits 4 and 5 is 0 instead of 1 in case SARADC resolution is 10-bit.

Offset: 0x060 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	THRH											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	THRL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 407. Watchdog Threshold Register *n* (WTHRHLR*n*)

Table 481. WTHRHLR*n* field descriptions

Field	Description
4:15 THRH	High threshold value for channel x
20:31 THRL	Low threshold value for channel x

39.5.1.12 Conversion Timing Register *n* (CTR*n*)

Offset: 0x094 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PRECHG				INPSAMP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 408. Conversion Timing Register *n* (CTR*n*)

Table 482. CTR*n* field descriptions

Field	Description
0 CRES	Conversion Resolution select This bit selects the conversion resolution for the conversion. 0 12-bit resolution in case of 12-bit SARADC. 10-bit resolution in case of 10-bit SARADC 1 10-bit resolution in case of 12-bit SARADC. 8-bit resolution in case of 10-bit SARADC
20:23 PRECHG	Precharging phase duration This bit field defines the precharging phase duration to be applied when this phase is enabled by PCE bit of channel data register. This duration is calculated as (PRECHG X 1/Frequency of SARADC clock). Refer to device datasheet for the minimum duration required for precharging phase.
24:31 INPSAMP	Sampling phase duration This bit field defines the sampling phase duration to be applied. This duration is calculated as (INPSAMP X 1/Frequency of SARADC clock). Refer to device datasheet for the minimum duration required for sampling phase.

39.5.1.13 Internal Channel Normal Conversion Mask Register *n* (ICNCMR*n*)

The normal conversion mask registers to channel association is described in *Table 484*.

Offset: 0x0A4 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 409. Internal Channel Normal Conversion Mask Register *n* (ICNCMR*n*)

Table 483. ICNCMR*n* field descriptions

Field	Description
0:31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 484. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 0:31
ICNCMR0	NCE_CH[31:0]

Table 484. Internal Channel Normal Conversion Mask Registers to Channel Association (continued)

Register	Register bits 0:31
ICNCMR1	NCE_CH[63:32]
ICNCMR2	NCE_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field NCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.14 Internal Channel Injected Conversion Mask Register *n* (ICJCMR*n*)

The injected conversion mask registers to channel association is described in *Table 486*.

Offset: 0x0B4 + n*0x4 (n = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 410. Internal Channel Injected Conversion Mask Register *n* (ICJCMR*n*)

Table 485. ICJCMR*n* field descriptions

Field	Description
0:31 JCE_CH[x]	Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 486. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 0:31
ICJCMR0	JCE_CH[31:0]
ICJCMR1	JCE_CH[63:32]
ICJCMR2	JCE_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field JCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.15 Power Down Exit Delay Register (PDEDR)

Offset: 0x0C8 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PDED							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 411. Power Down Exit Delay Register (PDEDR)

Table 487. PDEDR field descriptions

Field	Description
24:31 PDED	Power down exit delay duration Defines the delay between the power-down bit reset and the starting of conversion. The power-down delay is calculated as [PDED × 1/(SARADC clock frequency)]. The minimum required power down delay is the sum of $t_{ADCBIASINIT}$ (time required by ADCBIAS to generate correct value of bias current for all ADCs) and $t_{ADCINIT}$ (initialization time for ADC after the generation of bias current). Both values are provided in the device datasheet.

39.5.1.16 Internal Channel Data Register n (ICDR*n*)

The conversion results for each channel are loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid, timing parameter selection for each channel.

This register should be accessed via 32-bit R/W.

Offset: 0x100 + *n**0x4 (*n* = 0 to 95) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PCE	0	CTSEL		0	0	0	FCERR	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 412. Internal Channel Data Register *n* (ICDR*n*)

Table 488. ICDR*n* field descriptions

Field	Description
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled. 1 Precharge phase is enabled.
6:7 CTSEL	This bit field selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected. 01 CTR1 is selected. 10 CTR2 is selected. 11 CTR3 is selected.
11 FCERR	Fast Comparator Error Flag This bit is used to notify error condition on current fast comparator operation inside ADC analog block. It is automatically cleared when data is read. 0 Cleared by reading the Converted by software. 1 Fast Comparator Error Condition has occurred in current conversion.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten. 1 Previous converted data has been overwritten before having been read.

Table 488. ICDRn field descriptions (continued)

Field	Description
14:15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode. 01 Data is a result of Injected conversion mode. 10 Data is a result of CTU conversion mode. 11 Reserved
16:31 CDATA	Channel converted data **Note:** The data organization inside the CDATA field is dependent on: – The SARADC type which can either be a 12-bit type, a 10-bit type or a 9-bit type. – The setting of CTRx.CRES which selects if conversion is done with full resolution or with reduced resolution. – The setting of MCR.WLSIDE which selects the read data alignment. The valid data bits position is summarized in *Table 489*.

Table 489. Valid data bit organization

SARADC type	CTRx.CRES setting	MCR.WLSIDE setting	Effective Resolution	Data Alignment	Data Bit valid	Data Bit read as zero	Data Bit to be ignored
12-bit	0	0	12-bit	Right	CDATA[20:31]	CDATA[16:19]	—
		1		Left	CDATA[16:27]	CDATA[28:31]	—
	1	0	10-bit	Right	CDATA[22:31]	CDATA[16:21]	—
		1		Left	CDATA[16:25]	CDATA[26:31]	—
10-bit	0	0	10-bit	Right	CDATA[22:31]	CDATA[16:21]	—
		1		Left	CDATA[16:25]	CDATA[26:31]	—
	1	0	8-bit	Right	CDATA[24:31]	CDATA[16:23]	—
		1		Left	CDATA[16:23]	CDATA[24:31]	—
9-bit	0	0	9-bit	Right	CDATA[22:30]	CDATA[16:21]	CDATA[31]
		1		Left	CDATA[16:24]	CDATA[26:31]	CDATA[25]
	1	0	8-bit	Right	CDATA[24:31]	CDATA[16:23]	—
		1		Left	CDATA[16:23]	CDATA[24:31]	—

39.5.1.17 Watchdog Threshold Register *n* (WTHRHLR*n*)

Note: The number of WTHRHLR registers corresponds to the number of analogs watchdogs implemented in the ADC module.

The length of the threshold field depends on the SARADC resolution.

For 10-bit SARADC, the bits 4, 5, 20 and 21 are reserved.

Offset: 0x280 + n*0x4 (n = 4 to 15) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	THRH											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	THRL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 413. Watchdog Threshold Register *n* (WTHRHLR*n*)

Table 490. WTHRHLR*n* field descriptions

Field	Description
4:15 THRH	High threshold value for channel x
20:31 THRL	Low threshold value for channel x

39.5.1.18 Internal Channel Watchdog Select Register *n* (ICWSELR*n*)

These registers contain WSEL_CHx[0:3] bit fields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described in *Table 492*.

Offset: 0x2B0 + n*0x4 (*n* = 0 to 11) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL_CH[(8*n)+7]				WSEL_CH[(8*n)+6]				WSEL_CH[(8*n)+5]				WSEL_CH[(8*n)+4]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH[(8*n)+3]				WSEL_CH[(8*n)+2]				WSEL_CH[(8*n)+1]				WSEL_CH[8*n]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 414. Internal Channel Watchdog Select Register *n* (ICWSELR*n*)

Table 491. ICWSELR*n* field descriptions

Field	Description
0:3 WSEL_CH[(8*n)+7]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
4:7 WSEL_CH[(8*n)+6]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
8:11 WSEL_CH[(8*n)+5]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
12:15 WSEL_CH[(8*n)+4]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
16:19 WSEL_CH[(8*n)+3]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
20:23 WSEL_CH[(8*n)+2]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 491. ICWSELR*n* field descriptions (continued)

Field	Description
24:27 WSEL_CH[(8*n)+1]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
28:31 WSEL_CH[8*n]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 492. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register bits 0:31
ICWSELRn (n = 0 to 11)	WSEL_CH[(8*n)+7] … WSEL_CH[8*n]

The number of bits implemented for each WSEL_CHx depends on Watchdog number:

- If Watchdog number > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < Watchdog number <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < Watchdog number <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If Watchdog number = 1, only 1 bit is present.

Each nibble (WSEL_CHx) has a positional correspondence to the Internal Channel number (each channel has a 1:1 nibble correspondence), so each word ICWSELRx corresponds to eight internal Channels for a total of twelve 32-bit registers to covering all 96 test channels.

If an Internal channel is not defined, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on the Watchdog number. For example, if Watchdog number is 4 than only two LSBs of each WSEL_CHx are implemented and remaining two will remain unimplemented. If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

39.5.1.19 Internal Channel Watchdog Enable Register *n* (ICWENR*n*)

The Watchdog enable registers to channel association is described in *Table 494*.

Offset: 0x2E0 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 415. Internal Channel Watchdog Enable Register *n* (ICWENR*n*)

Table 493. ICWENRn field descriptions

Field	Description
0:31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 494. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register bits 0:31
ICWER0	WEN_CH[31:0]
ICWER1	WEN_CH[63:32]
ICWER2	WEN_CH[95:64]

Note: *If channel CH[x] is not existing then the corresponding bit field WEN_CH[x] is reserved and returns 0.*

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.20 Internal Channel Analog Watchdog Out of Range Register *n* (ICAWORR*n*)

The analog watchdog registers to channel association is described in *Table 496*.

Offset: 0x2F0 + n*0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 416. Internal Channel Analog Watchdog Out of Range Register *n* (ICAWORR*n*)

Table 495. ICAWORRn field descriptions

Field	Description
0:31 AWOR_CH[x]	Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Table 496. Internal Channel Analog Watchdog Registers to Channel Association

Register	Register bits 0:31
ICAWORR0	AWOR_CH[31:0]
ICAWORR1	AWOR_CH[63:32]
ICAWORR2	AWOR_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field AWOR_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.21 Test Channel Interrupt Pending Register (TCIPR)

The interrupt channel register to channel association is described in *Table 498*.

Offset: 0x400 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 417. Test Channel Interrupt Pending Register (TCIPR)

Table 497. TCIPR field descriptions

Field	Description
0:31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 498. TCIPR Register to Channel Association

Register	Register bits 0:31
TCIPR	EOC_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field EOC_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.22 Test Channel Interrupt Mask Register (TCIMR)

The interrupt mask register to channel association is described in *Table 500*.

Offset: 0x404 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 418. Test Channel Interrupt Mask Register (TCIMR)

Table 499. TCIMR field descriptions

Field	Description
0:31 IM_CH[x]	Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 500. Test Channel Interrupt Mask Register to Channel Association

Register	Register bits 0:31
TCIMR	IM_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field IM_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.23 Test Channel DMA Select Register (TCDSR)

The DMA select registers to channel association is described in *Table 502*.

Offset: 0x408 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 419. Test Channel DMA Select Register (TCDSR)

Table 501. TCDSR field descriptions

Field	Description
0:31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 502. Test Channel DMA Select Registers to Channel Association

Register	Register bits 0:31
TCDSR	DS_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field DS_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.24 Test Channel Normal Conversion Mask Register (TCNCMR)

The normal conversion mask register to channel association is described in *Table 504*.

Offset: 0x40C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 420. Test Channel Normal Conversion Mask Register (TCNCMR)

Table 503. TCNCMR field descriptions

Field	Description
0:31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 504. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register bits 0:31
TCNCMR	NCE_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field NCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.25 Test Channel Injected Conversion Mask Register (TCJCMR)

The injected conversion mask register to channel association is described in *Table 506*.

Offset: 0x410 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 421. Test Channel Injected Conversion Mask Register (TCJCMR)

Table 505. TCJCMR field descriptions

Field	Description
0:31 JCE_CH[x]	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 506. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register bits 0:31
TCJCMR	JCE_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field JCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.26 Test Channel Watchdog Select Register *n* (TCWSELR*n*)

This register contains WSEL_CHx[0:3] bit fields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select register to channel association is described in Table 508.

Offset: 0x414 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL_CH[(8*n)+103]				WSEL_CH[(8*n)+102]				WSEL_CH[(8*n)+101]				WSEL_CH[(8*n)+100]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH[(8*n)+99]				WSEL_CH[(8*n)+98]				WSEL_CH[(8*n)+97]				WSEL_CH[(8*n)+96]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 422. Test Channels Watchdog Select Register *n* (TCWSELR*n*)

Table 507. TCWSELR*n* field descriptions

Field	Description
4:7 WSEL_CH[(8*n)+103]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
0:3 WSEL_CH[(8*n)+102]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
8:11 WSEL_CH[(8*n)+101]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 507. TCWSELR*n* field descriptions (continued)

Field	Description
12:15 WSEL_CH[(8*n)+100]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
16:19 WSEL_CH[(8*n)+99]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
20:23 WSEL_CH[(8*n)+98]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
24:27 WSEL_CH[(8*n)+97]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
28:31 WSEL_CH[(8*n)+96]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 508. Test Channel Watchdog Select Registers to Channel Association

Register	Register bits 0:31
TCWSELR0	WSEL_CH[103] … WSEL_CH[96]
TCWSELR1	WSEL_CH[111] … WSEL_CH[104]
TCWSELR2	WSEL_CH[119] … WSEL_CH[112]
TCWSELR3	WSEL_CH[127] … WSEL_CH[120]

The number of bits implemented for each WSEL_CHx depends on Watchdog number:

- If Watchdog number > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < Watchdog number <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < Watchdog number <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If Watchdog number = 1, only 1 bit is present.

Each such nibble (WSEL_CHx) has positional correspondence to Test Channel number (each channel has a 1:1 nibble correspondence), so each word TCWSELRx corresponds to eight test Channels, thus requiring total four 32-bit registers to cover all 32 test channels.

If a Test channel is not defined, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on the Watchdog number. For example, if Watchdog number is 4 than only two LSBs of each WSEL_CHx are implemented and remaining two will remain unimplemented. If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

39.5.1.27 Test Channel Watchdog Enable Register (TCWENR)

The Watchdog enable registers to channel association is described in *Table 510*.

Offset: 0x424 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 423. Test Channel Watchdog Enable Register (TCWENR)

Table 509. TCWENR field descriptions

Field	Description
0:31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 510. Test Channel Watchdog Enable Registers to Channel Association

Register	Register bits 0:31
TCWENR	WEN_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field WEN_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.28 Test Channel Analog Watchdog Out of Range Register (TCAWORR)

The analog watchdog registers to channel association is described in *Table 512*.

Offset: 0x0428 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 424. Test Channel Analog Watchdog Out of Range Register (TCAWORR)

Table 511. TCAWORR field descriptions

Field	Description
0:31 AWOR_CH[x]	Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Table 512. Analog Watchdog Register to Channel Association

Register	Register bits 0:31
TCAWORR	AWOR_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field AWOR_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.29 Test Channel Connection with Analog Pin Register *n* (TCCAPR*n*)

Test channels connected through a 20 kΩ source impedance can be shorted with any internal analog channel using the configuration programed in these registers.

Test channel short registers to channel association is described in *Table 514*.

For analog test channel assignment, refer to the table SAR_ADC analog test channel assignment of chapter Analog-to-Digital Converters (ADC) Configuration.

Offset: 0x430 + *n**0x4 (*n* = 0 to 7) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ESIC_TCH[4*n+3]	ICSEL_TCH[4*n+3]							ESIC_TCH[4*n+2]	ICSEL_TCH[4*n+2]						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ESIC_TCH[4*n+1]	ICSEL_TCH[4*n+1]							ESIC_TCH[4*n]	ICSEL_TCH[4*n]						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 425. Test Channel Connection with Analog Pin Register *n* (TCCAPR*n*)

Table 513. TCCAPR*n* field descriptions

Field	Description
0 ESIC_TCH[4*n+3]	Enable short with internal channel for test channel x=4*n+3 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
1:7 ICSEL_TCH[4*n+3]	Internal channel selection for short with test channel x=4*n+3
8 ESIC_TCH[4*n+2]	Enable short with internal channel for test channel x=4*n+2 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
9:15 ICSEL_TCH[4*n+2]	Internal channel selection for short with test channel x=4*n+2
16 ESIC_TCH[4*n+1]	Enable short with internal channel for test channel x=4*n+1 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
17:23 ICSEL_TCH[4*n+1]	Internal channel selection for short with test channel x=4*n+1
24 ESIC_TCH[4*n]	Enable short with internal channel for test channel x=4*n 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
25:31 ICSEL_TCH[4*n]	Internal channel selection for short with test channel x=4*n

Table 514. Test Channel Short Registers to Channel Association

Register	Register bits 0:31 (arrangement of bit fields left to right)
TCCAPR0	ESIC_TCH3, ICSEL_TCH3, ESIC_TCH2, ICSEL_TCH2, ESIC_TCH1, ICSEL_TCH1, ESIC_TCH0, ICSEL_TCH0
TCCAPR1	ESIC_TCH7, ICSEL_TCH7, ESIC_TCH6, ICSEL_TCH6, ESIC_TCH5, ICSEL_TCH5, ESIC_TCH4, ICSEL_TCH4
TCCAPR2	ESIC_TCH11, ICSEL_TCH11, ESIC_TCH10, ICSEL_TCH10, ESIC_TCH9, ICSEL_TCH9, ESIC_TCH8, ICSEL_TCH8
TCCAPR3	ESIC_TCH15, ICSEL_TCH15, ESIC_TCH14, ICSEL_TCH14, ESIC_TCH13, ICSEL_TCH13, ESIC_TCH12, ICSEL_TCH12
TCCAPR4	ESIC_TCH19, ICSEL_TCH19, ESIC_TCH18, ICSEL_TCH18, ESIC_TCH17, ICSEL_TCH17, ESIC_TCH16, ICSEL_TCH16
TCCAPR5	ESIC_TCH23, ICSEL_TCH23, ESIC_TCH22, ICSEL_TCH22, ESIC_TCH21, ICSEL_TCH21, ESIC_TCH20, ICSEL_TCH20
TCCAPR6	ESIC_TCH27, ICSEL_TCH27, ESIC_TCH26, ICSEL_TCH26, ESIC_TCH25, ICSEL_TCH25, ESIC_TCH24, ICSEL_TCH24
TCCAPR7	ESIC_TCH31, ICSEL_TCH31, ESIC_TCH30, ICSEL_TCH30, ESIC_TCH29, ICSEL_TCH29, ESIC_TCH28, ICSEL_TCH28

39.5.1.30 Test Channel Data Register *n* (TCDR*n*)

The conversion results for each channel are loaded into data registers. Each data register contains:

- the conversion result
- status information related to ADC mode
- data valid
- timing parameter selection for each channel.

These registers are accessed 32-bit R/W.

Offset: 0x2D0 + *n**0x4 (*n* = 96 to 127) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PCE	0	CTSEL		0	0	0	FCERR	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 426. Test Channel Data Register *n* (TCDR*n*)

Table 515. TCDR*n* field descriptions

Field	Description
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled
6:7 CTSEL	This bit field selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected
11 FCERR	Fast Comparator Error Flag This bit is used to notify error condition on current fast comparator operation inside ADC analog block. It is automatically cleared when data is read. 0 Cleared by reading the Converted by software. 1 Fast Comparator Error Condition has occurred in current conversion.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software 1 Converted data is valid and has not been read yet
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14:15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16:31 CDATA	Channel converted data **Note:** If MCR[WLSIDE] is set to ‘0’ than the converted data is read right aligned with lower 12/10 bits representing the actual converted data (width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining upper 4/6 bits are read as zeros. Whereas if the MCR[WLSIDE] is set as ‘1’ than the converted data is read left aligned with upper 12/10 bits representing the actual converted data (width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining lower 4/6 bits are read as zeros.

39.5.1.31 External Channel Decode Signals Delay Register (ECDSDR)

Offset: 0x500 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	DSD											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 427. External Channel Decode Signals Delay Register (ECDSDR)

Table 516. ECDSDR field descriptions

Field	Description
20:31 DSD	Decode signals delay This bit field defines the delay between the external decode signals and the start of the sampling phase. It is used to take into account of the settling time required for the external mux. The decode signal delay is calculated as (DSD X 1/Frequency of SARADC clock).

39.5.1.32 External Channel Interrupt Pending Register *n* (ECIPR*n*)

The interrupt channel register to channel association is described in *Table 518*

Offset: 0x510 + n*0x4 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 428. External Channel Interrupt Pending Register *n* (ECIPR*n*)

Table 517. ECIPR*n* register field descriptions

Field	Description
0:31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 518. External Channel Interrupt Pending Registers to Channel Association

Register	Register bits 0:31
ECIPR0	EOC_CH[159:128]
ECIPR1	EOC_CH[191:160]
ECIPR2	EOC_CH[223:192]
ECIPR3	EOC_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field EOC_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.33 External Channel Interrupt Mask Register *n* (ECIMR*n*)

The interrupt mask register to channel association is described in *Table 520*.

Offset: 0x520 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 429. External Channel Interrupt Mask Register *n* (ECIMR*n*)

Table 519. ECIMR*n* field descriptions

Field	Description
0:31 IM_CH[x]	Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 520. External Channel Interrupt Mask Registers to Channel Association

Register	Register bits 0:31
ECIMR0	IM_CH[159:128]
ECIMR1	IM_CH[191:160]
ECIMR2	IM_CH[223:192]
ECIMR3	IM_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field IM_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.34 External Channel DMA Select Register *n* (ECDSR*n*)

The DMA select registers to channel association is described in *Table 522*.

Offset: 0x530 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 430. External Channel DMA Select Register *n* (ECDSR*n*)

Table 521. ECDSR*n* field descriptions

Field	Description
0:31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 522. DMA Select Registers to Channel Association

Register	Register bits 0:31
ECDSR0	DS_CH[159:128]
ECDSR1	DS_CH[191:160]
ECDSR2	DS_CH[223:192]
ECDSR3	DS_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field DS_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.35 External Channel Normal Conversion Mask Register *n* (ECNCMR*n*)

The normal conversion mask registers to channel association is described in *Table 524*.

Offset: 0x540 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 431. External Channel Normal Conversion Mask Register *n* (ECNCMR*n*)

Table 523. ECNCMR*n* field descriptions

Field	Description
0:31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 524. External Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 0:31
ECNCMR0	NCE_CH[159:128]
ECNCMR1	NCE_CH[191:160]
ECNCMR2	NCE_CH[223:192]
ECNCMR3	NCE_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field NCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.36 External Channel Injected Conversion Mask Register *n* (ECJCMR*n*)

The injected conversion mask registers to channel association are described in *Table 526*.

Offset: 0x550 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 432. External Channel Injected Conversion Mask Register *n* (ECJCMR*n*)

Table 525. ECJCMR*n* field descriptions

Field	Description
0:31 JCE_CH[x]	Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 526. External Channel Injected Conversion Mask Registers to Channel Association

Register	Register bits 0:31
ECJCMR0	JCE_CH[159:128]
ECJCMR1	JCE_CH[191:160]
ECJCMR2	JCE_CH[223:192]
ECJCMR3	JCE_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field JCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.37 External Channel Watchdog Select Register *n* (ECWSELR*n*)

These registers contain WSEL_CHx[0:3] bit fields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described in *Table 528*.

Offset: 0x0560 + n*0x4 (n = 0 to 15) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL_CH[(8*n)+135]				WSEL_CH[(8*n)+134]				WSEL_CH[(8*n)+133]				WSEL_CH[(8*n)+132]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH[(8*n)+131]				WSEL_CH[(8*n)+130]				WSEL_CH[(8*n)+129]				WSEL_CH[(8*n)+128]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 433. External Channel Watchdog Select Register *n* (ECWSELR*n*)

Table 527. ECWSELR*n* field descriptions

Field	Description
0:3 WSEL_CH[(8*n)+135]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
4:7 WSEL_CH[(8*n)+134]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
8:11 WSEL_CH[(8*n)+133]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
12:15 WSEL_CH[(8*n)+132]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
16:19 WSEL_CH[(8*n)+131]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 527. ECWSELRn field descriptions (continued)

Field	Description
20:23 WSEL_CH[(8*n)+130]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
24:27 WSEL_CH[(8*n)+129]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
28:31 WSEL_CH[(8*n)+128]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 528. External Channel Watchdog Enable Registers to Channel Association

Register	Register bits 0:31
ECWSELRn (n = 0 to 15)	WSEL_CH[(8*n)+135] ... WSEL_CH[(8*n)+128]

The number of bits implemented for each WSEL_CHx depends on Watchdog number:

- If Watchdog number > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < Watchdog number <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < Watchdog number <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If Watchdog number = 1, only 1 bit is present.

Each nibble (WSEL_CHx) has positional correspondence to the External Channel number (each channel has a 1:1 nibble correspondence), so each word ECWSELRx corresponds to eight external channels, thus requiring a total of sixteen 32-bit registers to cover all 128 external channels.

If an External channel is not defined, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on the Watchdog number. For example, if Watchdog number is 4, then only two LSBs of each WSEL_CHx are implemented and the remaining two will remain unimplemented. If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

39.5.1.38 External Channel Watchdog Enable Register *n* (ECWENR*n*)

The Watchdog enable registers to channel association is described in *Table 530*.

Offset: 0x5A0 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 434. External Channel Watchdog Enable Register *n* (ECWENR*n*)

Table 529. ECWENR*n* field descriptions

Field	Description
0:31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 530. External Channel Watchdog Enable Registers to Channel Association

Register	Register bits 0:31
ECWENR0	WEN_CH[159:128]
ECWENR1	WEN_CH[191:160]
ECWENR2	WEN_CH[223:192]
ECWENR3	WEN_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field WEN_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.39 External Channel Analog Watchdog Out of Range Register *n* (ECAWORR*n*)

The analog watchdog registers to channel association is described in *Table 532*.

Offset: 0x05B0 + n*0x4 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 435. External Channel Analog Watchdog Out of Range Register *n* (ECAWORR*n*)

Table 531. ECAWORR*n* field descriptions

Field	Description
0:31 AWOR_CH[x]	Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Table 532. External Channel Analog Watchdog Registers to Channel Association

Register	Register bits 0:31
ECAWORR0	AWOR_CH[159:128]
ECAWORR1	AWOR_CH[191:160]
ECAWORR2	AWOR_CH[223:192]
ECAWORR3	AWOR_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field AWOR_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

39.5.1.40 External Channel Mapping to Internal Channel Registers *n* (ECMICR*n*)

Each set of eight external channels can be mapped with any internal analog channel using the configuration programed in these registers.

Offset: 0x5C0 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	ICSEL_ECH[159+32*n]_[152+32*n]							0	ICSEL_ECH[151+32*n]_[144+32*n]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	ICSEL_ECH[143+32*n]_[136+32*n]							0	ICSEL_ECH[135+32*n]_[128+32*n]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 436. External Channel Mapping to Internal Channel Registers *n* (ECMICR*n*)

Table 533. ECMICR*n* field descriptions

Field	Description
1:7 ICSEL_ECH[159+32*n]_[152+32*n]	Internal channel selection for external channels (159+32*n) to (152+32*n)
9:15 ICSEL_ECH[151+32*n]_[144+32*n]	Internal channel selection for external channels (151+32*n) to (144+32*n)
17:23 ICSEL_ECH[143+32*n]_[136+32*n]	Internal channel selection for external channels (143+32*n) to (136+32*n)
25:31 ICSEL_ECH[135+32*n]_[128+32*n]	Internal channel selection for external channels (135+32*n) to (128+32*n)

Table 534. External Channel Mapping to Internal Channel Registers to Channel Association

Register	Register bits 0:31 (arrangement of bit fields left to right)
ECMICR0	ICSEL_ECH159_152, ICSEL_ECH151_144, ICSEL_ECH143_136, ICSEL_ECH135_128
ECMICR1	ICSEL_ECH191_184, ICSEL_ECH183_176, ICSEL_ECH175_168, ICSEL_ECH167_160
ECMICR2	ICSEL_ECH223_216, ICSEL_ECH215_208, ICSEL_ECH207_200, ICSEL_ECH199_192
ECMICR3	ICSEL_ECH255_248, ICSEL_ECH247_240, ICSEL_ECH239_232, ICSEL_ECH231_224

39.5.1.41 External Channel Data Register *n* (ECDR*n*)

The conversion results for each channel are loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid and some control information to select the required reference voltage, timing parameter selection for each channel.

This register should be accessed 32-bit R/W.

Offset: 0x3D0 + n*0x4 (n = 128 to 255) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PCE	0	CTSEL		0	0	0	FCERR	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 437. External Channel Data Register *n* (ECDR*n*)

Table 535. ECDR*n* field descriptions

Field	Description
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled. 1 Precharge phase is enabled.
6:7 CTSEL	This bit field selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected. 01 CTR1 is selected. 10 CTR2 is selected. 11 CTR3 is selected.
11 FCERR	Fast Comparator Error Flag This bit is used to notify error condition on current fast comparator operation inside ADC analog block. It is automatically cleared when data is read. 0 Cleared by reading the Converted by software. 1 Fast Comparator Error Condition has occurred in current conversion.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten. 1 Previous converted data has been overwritten before having been read.

Table 535. ECDRn field descriptions (continued)

Field	Description
14:15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode. 01 Data is a result of Injected conversion mode. 10 Data is a result of CTU conversion mode. 11 Reserved
16:31 CDATA	Channel converted data **Note:** If MCR[WLSIDE] is set to ‘0’ than the converted data is read right aligned with lower 12/10 bits representing the actual converted data (width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining upper 4/6 bits are read as zeros. Whereas if the MCR[WLSIDE] is set as ‘1’ than the converted data is read left aligned with upper 12/10 bits representing the actual converted data (width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining lower 4/6 bits are read as zeros.

39.6 Start of conversion pulse delay

The following table lists the delays between various trigger points (initiation request for conversion as registered at system clock) and the actual issue of start pulse (adc_start[(m)]) to the ADC hard-macrocell. Here, each possible trigger source is described with the number of clock cycles required for different operations until the trigger of adc_start. This table does not take into account any software delay involved in programming or polling of any bit in order to measure conversion timing.

Table 536. Start of conversion pulse delay table

Trigger source	Register control (PBRIDGE_ CLK)	Inter domain synchronization (PBRIDGE_CLK)	ADC state machine (SARADC_ CLK)	Inter domain synchronization (SARADC_CLK)	Comments
Normal conversion triggered by software (setting of NSTART of MCR)	1	—	2	1-2[(3)]	Cumulative delay from wen (byte_en) assertion to actual issue of adc_start pulse = 3-4 SARADC_CLK cycles + 1 PBRIDGE_CLK cycle.
Normal conversion triggered by external trigger (hardware trigger through)	—	—	2	1–2[(3)]	Cumulative delay from the triggering edge (rising or falling) to the actual issue of adc_start pulse = 3–4 SARADC_CLK cycles.

m. adc_start is the start of conversion indication to ADC hard-macrocell to start the conversion.

Table 536. Start of conversion pulse delay table (continued)

Trigger source	Register control (PBRIDGE_ CLK)	Inter domain synchronization (PBRIDGE_CLK)	ADC state machine (SARADC_ CLK)	Inter domain synchronization (SARADC_CLK)	Comments
Normal conversion within a chain	—	—	0 (from EOC(1) of previous conversion)	—	Delay from the EOC(1) of last conversion, The start of next conversion is issued exactly on the same cycle when EOC(1) is expected for previous conversion.
Injected conversion triggered by software (setting of JSTART bit of MCR)	1	—	3	1–2(3)	Cumulative delay from wen (byte_en) assertion. Here delay for first start pulse of the injected chain execution when FSM was originally in IDLE state is considered = 4–5 SARADC_CLK cycles + 1 PBRIDGE_CLK cycle.
Injected conversion triggered by external Trigger (hardware trigger)	—	—	3	1–2(3)	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 SARADC_CLK cycles.
Injected conversion within a chain (delay between consecutive conversion of injected chain)	—	—	0 (from EOC(1) of previous conversion)	—	Delay from EOC(1) of last injected channel and adc_start pulse of next injected channel. The start of next conversion is issued exactly on the same cycle when EOC(1) is expected for previous conversion
Injected conversion over normal conversion (software injected conversion chain)	1	1–2(3)	3–5(2)	1–2(3)	Cumulative delay from wen (byte_en) assertion to actual trigger of first injected channel, when the FSM was in conversion cycle of previous normal conversion = 4–7 SARADC_CLK cycles + 2-3 PBRIDGE_CLK cycles.

Table 536. Start of conversion pulse delay table (continued)

Trigger source	Register control (PBRIDGE_ CLK)	Inter domain synchronization (PBRIDGE_CLK)	ADC state machine (SARADC_ CLK)	Inter domain synchronization (SARADC_CLK)	Comments
Injection of conversion over normal conversion in JTRGSEQ mode	—	—	3	1–2(3)	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 SARADC_CLK cycles.
Abort of a Normal/Injected Channel conversion during Normal Chain/Injected Chain execution (setting of MCR[ABORT] bit)	1	1–2(3)	3–5(2)	1–2(3)	Delay from writing of ABORT bit (wen/byte_en assertion) to actual issue of adc_start pulse for the next channel conversion = 4–7 SARADC_CLK cycles + 2–3 PBRIDGE_CLK cycles.

1. EOC is the end of conversion from ADC hard macrocell
2. The above delay values are calculated over a range of system and ADC clock frequencies. With PBRIDGE_CLK(max) = 50 MHz to PBRIDGE_CLK(min) = 40 MHz and SARADC_CLK(max) = 16 MHz to SARADC_CLK(min) = 14 MHz. L.H.S value corresponds to max frequencies and RHS corresponds to min frequencies.
3. The above variation is owing to inter domain synchronization.

39.7 Initialization information

To initialize the SARADC registers for data conversion, the following programming sequence is required:

1. Program the PDEDR register with appropriate power down exit delay value and required CTU mode (MCR[CTU_MODE]) in case CTU mode has to be entered later on by setting CTUEN bit.
2. Exit the power down mode by clearing the MCR[PWDN] bit. After the power down mode is exited the SARADC enters the IDLE state, this can be checked by polling for MSR[ADCSTATUS] bits. Note that steps 3-7 can also be executed before step 2.
3. Program the required conversion mode and control options for example one-shot mode or scan mode, hardware trigger enable and selection for normal or injected conversions, hardware trigger edge select, overwrite enable, and so on, in MCR register.
4. Program the conversion mask registers ({I,T,E}C{N,J}CMR) for channels to be converted. If required, enable and select watchdog monitors ({I,T,E}CWENR, {I,T,E}CWSELR) as applicable for each channel, program the watchdog threshold values (WTHRHLR) for each watchdog monitor.
5. If required, Enable and select the External to Internal channel mapping by programming the ECMICR registers, Test to internal channel mapping by programming the TCCAPR registers.
6. Program the required Interrupt/DMA mask registers (DMAE, {I,T,E}CDSR, IMR, {I,T,E}C{N,J}MR, WTIMR) for each of the selected channels.

7. Program the Conversion timing registers(CTR0-3) for the required conversion durations. Enable the precharge phase, CTR selection per channel basis by programming the {I,T,E}CDR registers.
8. To initiate CTU mode conversion enter the selected CTU mode by setting the MCR[CTUEN] bit and program the CTU module accordingly to start the conversions. To initiate Normal/Injected mode conversions by software set the MCR[NSTART/JSTART] bit. To initiate hardware triggered Normal/Injected conversions program the external trigger module to provide the required triggers.
9. Read the conversion data and status in {I,T,E}CDR registers.

40 Temperature Sensor

40.1 Introduction

The device includes an onboard temperature sensor that monitors device temperature and delivers two analog outputs signals and three digital output signals.

The analog outputs consist of two voltage signals which vary linearly with the internal junction temperature: a voltage signal that is linearly increasing (PTAT: proportional to absolute temperature) and a voltage signal that is linearly decreasing (CTAT: complementary proportional to absolute temperature). The analog outputs are connected to an input channel of an ADC on the device. The internal junction temperature must be calculated by software based on the converted temperature values.

The three digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions. These signals notify the device to take action to appropriately adjust the device temperature in response to an out of specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

40.1.1 Signal Descriptions

This module has no external signals.

40.2 Functional description

The temperature sensor generates two analog outputs voltage, PTAT and CTAT which are proportional to the absolute current junction temperature of the device.

An on-chip ADC module is used to convert the analog outputs voltage, PTAT and CTAT, into a digital representation. These values, along with parameter values stored in onboard flash memory, are used by software to calculate the device junction temperature.

40.2.1 Linear temperature sensor (analog output generation)

The temperature sensor outputs two voltage proportional and complementary proportional to the internal junction temperature of the chip. These analog voltage signals are converted into digital values by an on-chip ADC. The temperature value is obtained from a linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programed into flash memory.

40.3 Temperature formula

40.3.1 Equations for converting TSENS voltages to junction temperature

Table 537. Calibration constants [1] [2]

Constant	Description
P1	Code from the ADC converting PTAT output voltage at 150 °C
P2	Code from the ADC converting PTAT output voltage at -40 °C
C1	Code from the ADC converting CTAT output voltage at 150 °C
C2	Code from the ADC converting CTAT output voltage at -40 °C

1. It is mandatory to keep the ADC reference constant while measuring P1, P2, C1, and C2
2. Flash locations for P2, C2, P1 and C1 are respectively 0x400000, 0x400002, 0x400004 and 0x400006.

In the equations below:

- Pn and Cn are the calibration constants described in *Table 537*.
- T is the unknown device temperature in °C.
- Px is the code from the ADC converting PTAT output voltage at the temperature T with any ADC reference voltage Vref
- Cx is the code from the ADC converting CTAT output voltage at the temperature T with the same ADC reference voltage Vref.
- T2 = -40 °C
- T1 = 150 °C

Equation 24 $A = Px \cdot C2 - (Cx \cdot P2)$

Equation 25 $B = Cx \cdot P1 - (Px \cdot C1)$

The temperature T is calculated as

Equation 26 $T = T2 + \frac{A \cdot (T1 - T2)}{A + B}$

Caution: It is mandatory to have the same value as the ADC reference, while measuring Px and Cx.

40.3.2 Equations for converting TSENS voltages into constant reference (Digital Bandgap Voltage)

By converting the two outputs of the Temperature Sensor through ADC, one can find out a constant reference code (REFCODE) which does not change with temperature.

- Pn and Cn are the calibration constants described in *Table 537*.
- Px is the code from the ADC converting PTAT output voltage at any temperature T with any ADC reference voltage Vref.
- Cx is the code from the ADC converting CTAT output voltage at the same temperature T with the same ADC reference voltage Vref.

Equation 27

$$REFCODE = Px + \left(\frac{(P2 - P1)}{(C1 - C2)}\right) \times Cx$$

Caution: It is mandatory to measure Px and Cx at the same ADC reference voltage and at the same temperature.

Note: Software calculation must be made with negligible precision loss associated with limited bit representation of intermediate calculated results.

41 Body Cross Triggering Unit (BCTU)

41.1 Introduction

The Body Cross Triggering Unit (BCTU) allows to synchronize an ADC conversion with a timer event from eMIOS (every mode which can generate a DMA request can trigger BCTU), or PIT_RTI. To select which ADC channel must be converted on a particular timer event, the BCTU provides the ADC with a 7-bit channel number. This channel number can be configured for each timer channel event by the application.

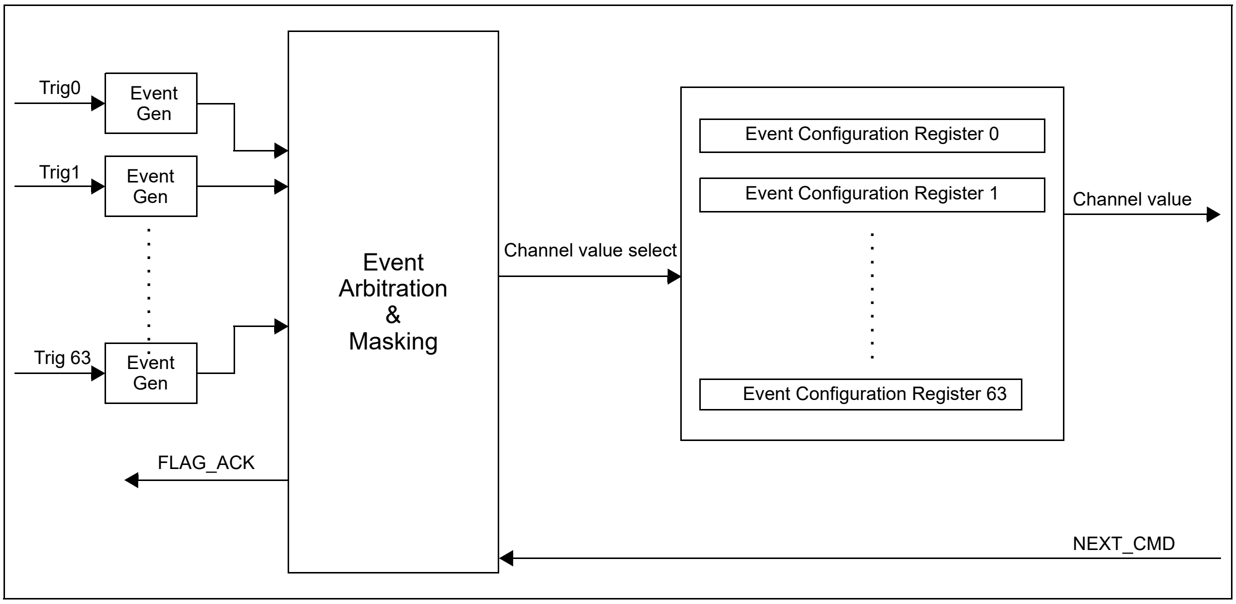
41.2 Main features

- Single cycle delayed trigger output. The trigger output is a combination of 64 (generic value) input flags/events connected to different timers in the system.
- One event configuration register dedicated to each timer event allows to define the corresponding ADC channel.
- Acknowledgment signal to eMIOS/PIT_RTI for clearing the flag
- Synchronization with ADC to avoid collision

41.3 Block diagram

The BCTU block diagram is shown in *Figure 438*.

Figure 438. Body Cross Triggering Unit block diagram



41.4 Memory map and register descriptions

The BCTU registers are listed in *Table 538*.

Table 538. BCTU memory map

Offset	Register	Location
0x000	Control Status Register (BCTU_CSR)	Section 41.4.1
0x004–0x02C	Reserved	
n = 0 to 63		
0x030 + n*0x4	Event Configuration Register *n* (BCTU_EVTCFGR*n*)	Section 41.4.2

41.4.1 Control Status Register (BCTU_CSR)

Offset: 0x000 Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	TRGIEN(1)	TRGI	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. BCTU is not able to trigger interrupt to CPU because interrupt signal is not routed to the interrupt controller (INTC).

Figure 439. Control Status Register (BCTU_CSR)

Table 539. BCTU_CSR field descriptions

Field	Description
24 TRGIEN	Trigger Interrupt Request Enable 0 Trigger interrupt request disabled 1 Trigger interrupt request enabled. A request is generated if the TRGI flag is set.
25 TRGI	Trigger Interrupt Flag This flag is set by hardware when the trigger output request is generated after a valid input event is detected. It is cleared by software. 0 No trigger output request 1 Trigger output request is generated

41.4.2 Event Configuration Register *n* (BCTU_EVTCFGR*n*)

Offset: 0x030 + n*0x4 (n=0 to 63) Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TM	CLR_FLAG(1)	0	0	0	0	0	0	0	CHANNEL_VALUE						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit implementation is generic based and implemented only for inputs mapped to PIT_RTI event flags.

Figure 440. Event Configuration Register *n* (BCTU_EVTCFGR*n*)

Table 540. BCTU_EVTCFGR*n* field descriptions

Field	Description
16 TM	Trigger Mask 0 Trigger masked 1 Trigger enabled
17 CLR_FLAG	To provide flag_ack through software 1 Flag_ack is forced to '1' for the particular event 0 Flag_ack is dependent on flag servicing
25:31 CHANNEL_VALUE	0xxxxxx ADC internal channel number from 0 to 63 1xxxxxx ADC external channel number from 128 to 191 **Note:** When bit 25 = 1 an offset of 128 is added to the value defined by bits 26:31 : that allows to address external channels inside ADC module.

These registers contain the ADC channel number to be converted when the timer event occurs.

When CLR_FLAG is set, BCTU ignores any event from PIT_RTI if some triggered conversion is going on in ADC (conversion from BCTU to ADC). But if BCTU is not busy in sending any trigger to ADC, then it will send the trigger to ADC for conversion of configured channel in BCTU_EVTCFGR register. In that case, setting CLR_FLAG bit would not provide flag_ack through software.

The CLR_FLAG bit has to be used cautiously as setting this bit may result in a loss of events.

The event input can be masked by writing '0' to bit TM of the BCTU_EVTCFGR register. Writing '1' to bit TM enables the BCTU triggering and automatically disables the DMA connection for the corresponding eMIOS channel.

Note: The BCTU tracks issued conversion requests to the ADC. When the ADC is being triggered by the BCTU and there is a need to shut down the ADC, the ADC must be allowed to complete conversions before being shut down. This ensures that the BCTU is notified of completion; if the ADC is shut down while performing a BCTU-triggered conversion, the BCTU is not notified and will not be able to trigger further conversions until the device is reset.

41.5 Functional description

This peripheral is used to synchronize ADC conversions with timer events (from eMIOS or PIT_RTI). When a timer event occurs, the BCTU triggers an ADC conversion providing the ADC channel number to be converted. In case concurrent events occur the priority is managed according to the index of the timer event. The trigger output is a single cycle pulse used to trigger ADC conversion of the channel number provided by the BCTU.

Each trigger input from the BCTU is connected to the Event Trigger signal of an eMIOS channel. The assignment between eMIOS outputs and BCTU trigger inputs of the device is defined in the *Chapter 7: Device configuration*.

Each event has a dedicated configuration register (BCTU_EVTCFGR). These registers store a channel number which is used to communicate which channel needs to be converted.

In case several events are pending for ADC request, the priority is managed according to the timer event index. The lowest index has the highest priority. Once an event has been serviced (conversion requested to ADC) the eMIOS flag is cleared by the BCTU and next prior event is handled.

The acknowledgment signal can be forced to ‘1’ by setting the CLR_FLAG bit of the BCTU_EVTCFGR register. These bits are implemented for only those input flags to which PIT_RTI flags are connected. Providing these bits offers the option of clearing PIT_RTI flags by software.

42 System Timer Module (STM)

42.1 Introduction

This section provides an overview, list of features, and modes of operation for the STM.

Note: For chip-specific implementation details of this module's instances, see the chip configuration information.

42.1.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the HPBM_CLK divided by an 8-bit prescale value (1 to 256).

42.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

42.1.3 Modes of operation

The STM supports two device modes of operation: Normal and Debug. When the STM is enabled in Normal mode, its counter runs continuously. In Debug mode, operation of the counter is controlled by the FRZ bit in the STM Control Register (STM_CR). If STM_CR[FRZ] is set, the counter is stopped in Debug mode, otherwise it continues to run.

42.2 External signal description

The STM does not have any external interface signals.

42.3 Memory map and register definition

The STM programming model includes a group of 32-bit registers: a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

42.3.1 Memory map

The STM memory map is shown in *Table 541*.

Table 541. STM memory map

Address offset	Register	Location
0x0000	STM Control Register (STM_CR)	*Section 42.3.2.1*
0x0004	STM Counter Value (STM_CNT)	*Section 42.3.2.2*
0x0008–0x000B	Reserved	
0x000C–0x000F	Reserved	
STM channel registers per channel (n=0 to 3)		
0x0010 + n*0x10	STM Channel n Control Register (STM_CCRn)	*Section 42.3.2.3*
0x0014 + n*0x10	STM Channel n Interrupt Register (STM_CIRn)	*Section 42.3.2.4*
0x0018 + n*0x10	STM Channel n Compare Register (STM_CMPn)	*Section 42.3.2.5*
0x001C + n*0x10	Reserved	
0x004C–0x3FFF	Reserved	

42.3.2 Register descriptions

42.3.2.1 STM Control Register (STM_CR)

The STM Control Register (STM_CR) includes the prescale value, freeze control and timer enable bits.

Offset: 0x000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPS								0	0	0	0	0	0	FRZ	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 441. STM Control Register (STM_CR)

Table 542. STM_CR field descriptions

Field	Description
16:23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide HPBM_CLK by 1 0x01 Divide HPBM_CLK by 2 ... 0xFF Divide HPBM_CLK by 256
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. **Note:** When the microcomputer enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode. 0 STM counter continues to run in debug mode 1 STM counter is stopped in debug mode
31 TEN	Timer Counter Enabled. 0 Counter is disabled 1 Counter is enabled

42.3.2.2 STM Count Register (STM_CNT)

The STM Count Register (STM_CNT) holds the timer count value.

Offset: 0x004 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 442. STM Count Register (STM_CNT)

Table 543. STM_CNT field descriptions

Field	Description
0:31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the HPBM_CLK divided by the prescale value.

42.3.2.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCR*n*) has the enable bit for channel n of the timer.

Offset: 0x10 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 443. STM Channel Control Register (STM_CCR*n*)

Table 544. STM_CCR*n* field descriptions

Field	Description
31 CEN	Channel Enable. 0 The channel is disabled 1 The channel is enabled

42.3.2.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register (STM_CIR*n*) has the interrupt flag for channel n of the timer.

Offset: 0x14 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 444. STM Channel Interrupt Register (STM_CIR*n*)

Table 545. STM_CIR*n* field descriptions

Field	Description
31 CIF	Channel Interrupt Flag 0 No interrupt request 1 Interrupt request due to a match on the channel

42.3.2.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Offset: 0x18 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 445. STM Channel Compare Register (STM_CMP*n*)

Table 546. STM_CMP*n* field descriptions

Field	Description
0:31 CMP	Compare value for channel n. If the STM_CCR*n*[CEN] bit is set and the STM_CMP*n* register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIR*n*[CIF] bit is set.

42.4 Functional description

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the HPBM_CLK frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode the counter operation is controlled by the STM_CR[FRZ] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode, otherwise it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*) and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit. When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing a 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

Note: The STM counter does not advance when the HPBM_CLK is stopped.

43 Software Watchdog Timer (SWT)

43.1 Introduction

This section provides an overview, list of features, and modes of operation for the SWT.

Note: For chip-specific implementation details of this module's instances, refer to the chip configuration information.

43.1.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out.

Note: The SWT runs over all device mode except the STANDBY mode where it is not powered.

43.1.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

43.1.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR *Section 43.2.1.1: SWT Control Register (SWT_CR)*. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit of the SWT_CR register. Refer to *Section 43.2.1.1: SWT Control Register (SWT_CR)*. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

43.2 Memory map and register definition

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access, otherwise a bus error is generated. If either the HLK or

SLK bits in the SWT_CR are set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

The SWT memory map is shown in *Table 547*.

Table 547. SWT memory map

Offset	Register	Location
0x0000	SWT Control Register (SWT_CR)	*Section 43.2.1.1*
0x0004	SWT Interrupt Register (SWT_IR)	*Section 43.2.1.2*
0x0008	SWT Time-Out Register (SWT_TO)	*Section 43.2.1.3*
0x000C	SWT Window Register (SWT_WN)	*Section 43.2.1.4*
0x0010	SWT Service Register (SWT_SR)	*Section 43.2.1.5*
0x0014	SWT Counter Output Register (SWT_CO)	*Section 43.2.1.6*
0x0018	SWT Service Key Register (SWT_SK)	*Section 43.2.1.7*
0x001C–0x3FFF	Reserved	

43.2.1 Register descriptions

The following sections detail the individual registers within the SWT programming model.

Note: Refer to the memory map for the base address of SWT registers.

43.2.1.1 SWT Control Register (SWT_CR)

The SWT_CR contains fields for configuring and controlling the SWT. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0	0	0	0	0	0	0	0
W																
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	SMD		RIA	WND	ITR	HLK	SLK	1	STP	FRZ	WEN
W																
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

1. The reset value for the SWT_CR is device specific. Refer to the device configuration chapter.

Figure 446. SWT Control Register (SWT_CR)

Table 548. SWT_CR field descriptions

Field	Description
0:7 MAP[0:7]	Master Access Protection for Master n The platform bus master assignments are device specific. Not all MAPn fields are implemented. Refer to the device configuration chapter for master ports that are implemented on the SMPU. 0 Access for the master is not enabled 1 Access for the master is enabled
21:22 SMD	Service Mode 00 Fixed Service Sequence, the watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to the SWT_SR. 01 Keyed Service Sequence, the watchdog is serviced by writing two pseudo-random key values to the SWT_SR. 10 Fixed Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register which can not be updated while the watchdog is enabled. Support for this mode is device and instance specific. 11 Incremental Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register which can be updated. Support for this mode is device and instance specific.
23 RIA	Reset on Invalid Access 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a system reset if WEN=1
24 WND	Window Mode 0 Regular mode, service sequence can be done at any time 1 Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
25 ITR	Interrupt Then Reset 0 Generate a reset on a time-out 1 Generate an interrupt on an initial time-out, reset on a second consecutive time-out
26 HLK	Hard Lock This bit is only cleared at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO SWT_WN and SWT_SK are read/write registers if HLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
29 STP	Stop Mode Control Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled 0 SWT is disabled 1 SWT is enabled

43.2.1.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Offset: 0x0004 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 447. SWT Interrupt Register (SWT_IR)

Table 549. SWT_IR field descriptions

Field	Description
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request. 1 Interrupt request due to an initial time-out.

43.2.1.3 SWT Time-Out Register (SWT_TO)

The SWT Time-Out (SWT_TO) register contains the 32-bit time-out period. The reset value for this register is device specific. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0008 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	WTO																															
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

1. The reset value of the SWT_TO register is device specific. Refer to the device configuration chapter.

Figure 448. SWT Time-Out Register (SWT_TO)

Table 550. SWT_TO field descriptions

Field	Description
0:31 WTO	Watchdog time-out period in clock cycles An internal 32-bit down counter is loaded with this value when the service sequence is written or when the SWT is enabled. The initial value is 0x0003_FDE0. The minimum value is 0x0000_0100.

43.2.1.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x000C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 449. SWT Window Register (SWT_WN)

Table 551. SWT_WN field descriptions

Field	Description
0:31 WST	Window start value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

43.2.1.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Offset: 0x0010 Access: Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
W																	WSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 450. SWT Service Register (SWT_SR)

Table 552. SWT_SR field descriptions

Field	Description
16:31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the soft lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] = 01, two pseudo-random key values are written to service the watchdog, refer to *Section 43.2.1.1: SWT Control Register (SWT_CR)* for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

43.2.1.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Offset: 0x0014 Access: Read-Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT																															
W																																
Reset[1]	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

1. SWT_CO will reflect the internal value of the counter when WEN=0, and will be read as 0x0000_0000 when WEN=1

Figure 451. SWT Counter Output Register (SWT_CO)

Table 553. SWT_CO field descriptions

Field	Description
0:31 CNT	Watchdog Count When the watchdog is disabled (SWT_CR[WEN]=0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN]=1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

43.2.1.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0018 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 452. SWT Service Key Register (SWT_SK)

Table 554. SWT_SK field descriptions

Field	Description
16:31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] = 01, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^16.

43.3 Functional description

43.3.1 Introduction

The SWT is a 32-bit timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- A control register (SWT_CR)
- An interrupt register (SWT_IR)
- A time-out register (SWT_TO)
- A window register (SWT_WN)
- A service register (SWT_SR)
- A counter output register (SWT_CO)
- A service key register (SWT_SK)

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

43.3.1.1 SWR_CR register

The SWT_CR register includes bits to enable the timer, set configuration options and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific[n]. If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

43.3.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100 in which case the time-out period is set to 0x100.When the SWT is enabled, this time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed.The source used to drive the down counter and the reset value of the SWT_TO register is device specific, refer to the device configuration chapter.

43.3.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

n. Refer to SWT section in device configuration chapter.

43.3.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

43.3.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

43.3.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

43.3.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

43.3.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation which is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- Fixed service sequence
- Keyed service sequence
- Fixed execution address
- Incremental execution address

43.3.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected which requires writing 0xA602, 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

43.3.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected which requires writing two pseudo-random keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudo-random key generator defined in *Equation 28*. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

Equation 28

$$SK_{n+1} = (17*SK_n+3) \bmod 2^{16}$$

43.3.4.3 Fixed execution address mode

If the SWT_CR[SMD] field is set to 10b, the fixed execution address mode is selected which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register cannot be updated while the watchdog is enabled.

43.3.4.4 Incremental execution address mode

If the SWT_CR[SMD] field is set to 11b, the incremental execution address mode is selected in which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can be updated while the watchdog is enabled.

43.3.4.5 Window mode

If window mode is enabled (SWT_CR[WND] bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_CR[RIA] bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to ‘1000’ then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

43.3.5 Time-out

The SWT_CR[ITR] (“Interrupt Then Reset” bit) controls the action taken when a time-out occurs. If the SWT_CR[ITR] bit is not set, a reset is generated immediately on a time-out. If the SWT_CR[ITR] bit is set, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period. If the service sequence is not written before the second consecutive time-out, the SWT generates a system reset. The interrupt is indicated by the time-out interrupt flag (SWT_IR[TIF]). The interrupt request is cleared by writing a one to the SWT_IR[TIF] bit.

43.3.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

44 Periodic Interrupt Timer (PIT)

44.1 Introduction

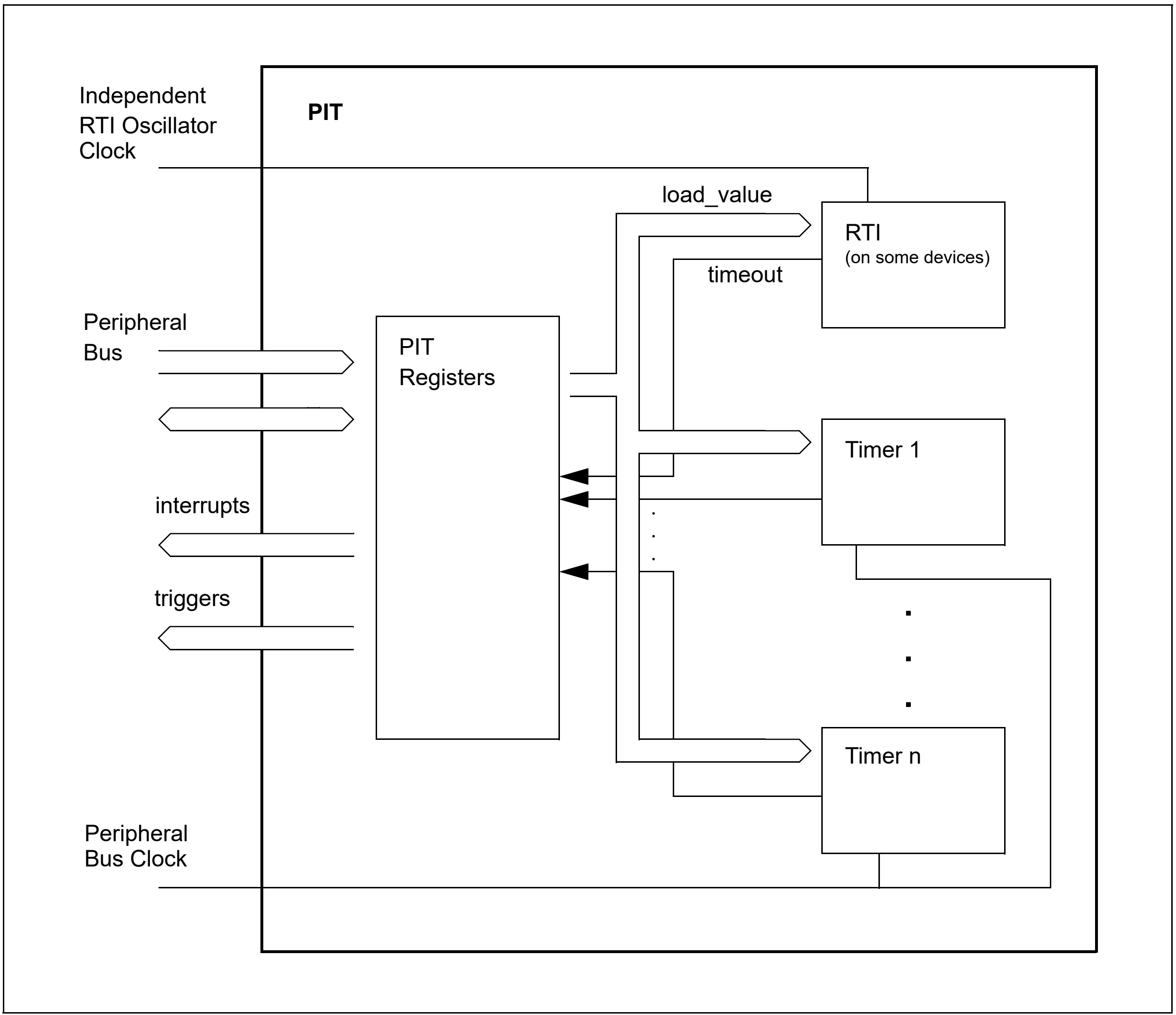
The PIT timer module is an array of timers that can be used to raise interrupts and trigger DMA channels. It includes a dedicated Real Time Interrupt Timer (RTI), which runs on a separate clock and can be used for system wakeup.

Note: For the chip-specific implementation details of this module's instances refer to the Device Configuration chapter.

44.1.1 Block diagram

Figure 453 shows the PIT block diagram.

Figure 453. Block diagram of the PIT



Note: Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

44.1.2 Features

The main features of this block are:

- One RTI (Real-Time Interrupt) timer to wake up the CPU in Stop mode
- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable
- RTI interrupt can be raised, even when the bus clock is switched off
- Power saving with a separate input clock for the RTI timer (all other timers share one common core clock)
- Independent timeout periods for each timer

44.2 Signal description

This module has no external pins.

44.2.1 Memory map/register description

This section provides a detailed description of all registers accessible in this module.

Note: Reserved registers will read as 0, writes will have no effect.

Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

The RTI registers should be programed only when the RTI clock is running.

Table 555. PIT memory map

Offset (hex)	Register name	Location
0x0000	PIT Control Register (MCR)	Section 44.2.1.1
0x0004–0x00DF	Reserved	
0x00E0	PIT Upper Lifetime Timer Register (LTMR64H)	Section 44.2.1.2
0x00E4	PIT Lower Lifetime Timer Register (LTMR64L)	Section 44.2.1.3
0x00E8–0x00EF	Reserved	
0x00F0	PIT RTI Timer Load Value Register (RTI_LDVAL)	Section 44.2.1.4
0x00F4	PIT RTI Current Timer Value Register (RTI_CVAL)	Section 44.2.1.5
0x00F8	PIT RTI Timer Control Register (RTI_TCTRL)	Section 44.2.1.6
0x00FC	PIT RTI Timer Flag Register (RTI_TFLG)	Section 44.2.1.7
PIT registers per channel (n = 0 to 7)		
0x0100 + *n**0x10	PIT Timer Load Value Register n (LDVAL*n*)	Section 44.2.1.8
0x0104 + *n**0x10	PIT Current Timer Value Register n (CVAL*n*)	Section 44.2.1.9
0x0108 + *n**0x10	PIT Timer Control Register n (TCTRL*n*)	Section 44.2.1.10

Table 555. PIT memory map (continued)

Offset (hex)	Register name	Location
0x010C + *n**0x10	PIT Timer Flag Register n (TFLG*n*)	Section 44.2.1.11
0x0180–0x3FFF	Reserved	

44.2.1.1 PIT Module Control Register (MCR)

The PIT Module Control Register (MCR) controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MDIS_RTI	MDIS	FRZ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 454. PIT Module Control Register (MCR)

Table 556. MCR field descriptions

Field	Description
29 MDIS_RTI	Module Disable - RTI section This is used to disable the RTI timer. This bit must be enabled before any RTI setup is done. 0 Clock for RTI timers is enabled. 1 Clock for RTI timers is disabled.
30 MDIS	Module Disable This is used to disable the module clock. The RTI timer is not affected by this bit. This bit must be enabled before any other setup is done. 0 Clock for PIT timers is enabled. 1 Clock for PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters debug mode. 0 Timers continue to run in debug mode. 1 Timers are stopped in debug mode.

44.2.1.2 PIT Upper Lifetime Timer Register (LTMR64H)

The PIT Upper Lifetime Timer Register (LTMR64H) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

Offset: 0x00E0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LTH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 455. PIT Upper Lifetime Timer Register (LTMR64H)

Table 557. LTMR64H field descriptions

Field	Description
0:31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

44.2.1.3 PIT Lower Lifetime Timer Register (LTMR64L)

The PIT Lower Lifetime Timer Register (LTMR64L) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Offset: 0x00E4 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LTL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 456. PIT Lower Lifetime Timer Register (LTMR64L)

Table 558. LTMR64L field descriptions

Field	Description
0:31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

44.2.1.4 PIT RTI Timer Load Value Register (RTI_LDVAL)

This register selects the timeout period for the timer interrupts.

In case of the RTI, it takes several cycles until this value is synchronized into the RTI clock domain. For all other timers the value change is visible immediately. The synchronization mechanism allows 0 wait states for RTI in this case.

Offset: 0x00F0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 457. PIT RTI Timer Load Value Register (RTI_LDVAL)

Table 559. RTI_LDVAL field descriptions

Field	Description
0:31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. **Note:** The RTI timer must not be set to a value lower than 32 cycles, otherwise interrupts may be lost, as it takes several cycles to clear the RTI interrupt. For the other timers, this limit does not apply, however there will be practical limits, since the processor will require several cycles to service an interrupt.

44.2.1.5 PIT RTI Current Timer Value Register (RTI_CVAL)

This register indicates the current timer position.

For the RTI timers, it shows a value which is several cycles old, since it originates from a potentially different clock domain.

Offset: 0x00F4 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 458. PIT RTI Current Timer Value Register (RTI_CVAL)

Table 560. RTI_CVAL field descriptions

Field	Description
0:31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. **Note:** The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.

44.2.1.6 PIT RTI Timer Control Register (RTI_TCTRL)

This register contains the control bits for each timer.

Offset: 0x00F8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 459. PIT RTI Timer Control Register (RTI_TCTRL)

Table 561. RTI_TCTRL field descriptions

Field	Description
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

44.2.1.7 PIT RTI Timer Flag Register (RTI_TFLG)

This register holds the PIT interrupt flags.

Offset: 0x00FC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 460. PIT RTI Timer Flag Register (RTI_TFLG)

Table 562. RTI_TFLG field descriptions

Field	Description
31 TIF	Timer Interrupt Flag TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

44.2.1.8 PIT Timer Load Value Register *n* (LDVAL*n*)

Offset: 0x0100 + n * 0x10 (n = 0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 461. PIT Timer Load Value Register *n* (LDVAL*n*)

Table 563. LDVAL*n* field descriptions

Field	Description
0:31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

44.2.1.9 PIT Current Timer Value Register *n* (CVAL*n*)

Offset: 0x0104 + *n**0x10 (n = 0 to 7) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 462. PIT Current Timer Value Register *n* (CVAL*n*)

Table 564. CVAL*n* field descriptions

Field	Description
0:31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. **Note:** The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.

44.2.1.10 PIT Timer Control Register *n* (TCTRL*n*)

Offset: 0x0108 + *n**0x10 (*n* = 0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 463. Timer Control Register (TCTRL*n*)

Table 565. TCTRL*n* field descriptions

Field	Description
29 CHN	Chain Mode Bit When activated, timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer (example, for channel 2 if this bit is set timer2 is chained 1).
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

44.2.1.11 PIT Timer Flag Register *n* (TFLG*n*)

Offset: 0x010C + *n**0x10 (*n* = 0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 464. Timer Flag Register *n* (TFLG*n*)

Table 566. TFLG*n* field descriptions

Field	Description
31 TIF	Timer Interrupt Flag TIF is set to '1' at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

44.3 Functional description

This section provides the functional description of the module.

44.3.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts. Each interrupt is available on a separate interrupt line.

44.3.1.1 Timers/RTI

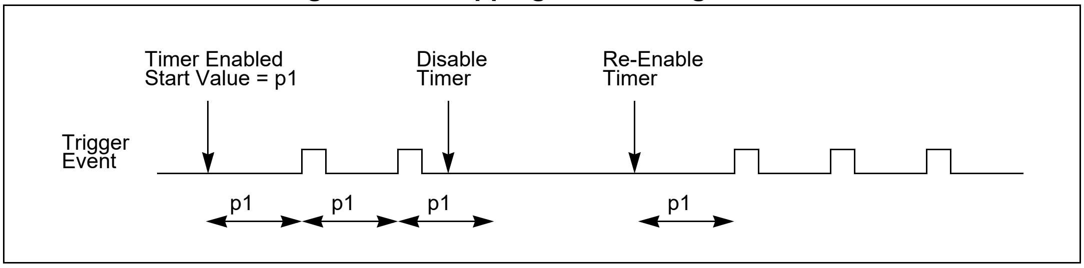
The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared. Since in the case of the RTI, clearing the interrupt crosses clock domains, a minimum load value of 32 should be maintained.

If desired, the current counter value of the timer can be read via the CVAL registers. The value of the RTI counter can be delayed considerably, as it is synchronized to the bus clock from the RTI clock domain.

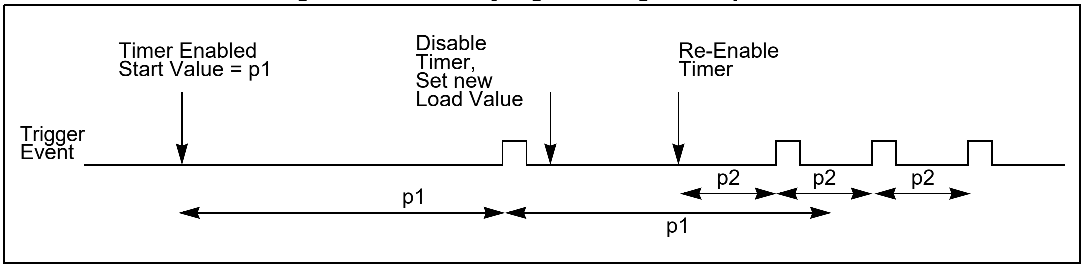
The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (refer to the following figure).

Figure 465. Stopping and starting a timer



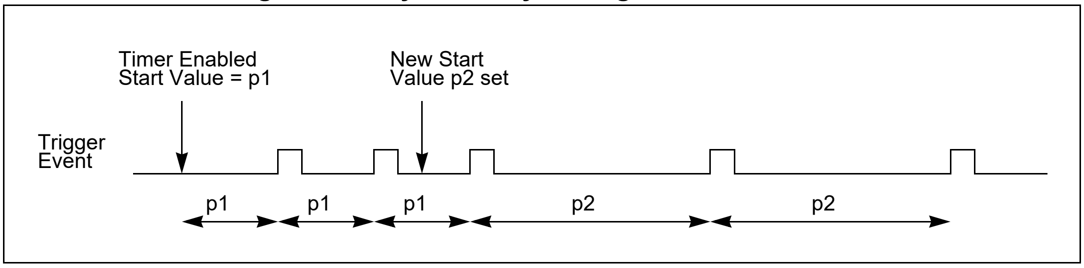
The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (refer to the following figure).

Figure 466. Modifying running timer period



It is also possible to change the counter period without restarting the timer by writing the LDVAL register with the new load value. This value will then be loaded after the next trigger event (refer to the following figure).

Figure 467. Dynamically setting a new load value



44.3.1.2 Debug mode

In debug mode, the timers will be frozen based on field MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (for example, the timer values) and then continue the operation.

44.3.2 Interrupts

All the timers support interrupt generation. The RTI is typically used for system wakeup, but can be used for interrupt generation as well. Refer to the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting the TIE bits. The timer interrupt flags (TIF) are set to ‘1’ when a timeout occurs on the associated timer, and are cleared to ‘0’ by writing a ‘1’ to that TIF bit.

The PIT RTI generates a real time interrupt when the selected interrupt time period elapses. The RTI interrupt is disabled locally by setting the TIE bit to zero. The real time interrupt flag (TIF) is set to ‘1’ when a timeout occurs, and is cleared by writing a ‘1’ to the TIF bit. (The flag will be set regardless whether the interrupt is enabled.)

The RTI can be used for periodic wakeup from a low power mode. It can also be used to generate a general purpose interrupt.

44.3.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer *n*-1 has counted down to 0, counter *n* will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

44.4 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- The RTI clock has a frequency of 10 MHz.
- The RTI creates a wakeup interrupt every 500 ms.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

First the PIT module must be activated by writing a 0 to field MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns and the 10 MHz frequency equates to a clock period of 100 ns. Therefore, the RTI timer needs to trigger every 500 ms/100 ns = 5000000 cycles.

Timer 1 needs to trigger every 5.12 ms/20 ns = 256000 cycles and timer 3 every 30 ms/20 ns = 1500000 cycles.

The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) - 1

This means RTI_LDVAL will be written with 004C4B3F hex, LDVAL1 should be written with 0x0003E7FF, and LDVAL3 should be written with 0x0016E35F.

To generate the wakeup interrupt, the interrupt line must be enabled by writing a ‘1’ to bit RTI_TCTRL[TIE]. To start the RTI, bit RTI_TCTRL[TEN] must also be set.

The interrupt for Timer 1 is enabled by setting bit TCTRL1[TIE]. The timer is started by writing ‘1’ to bit TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore Timer 3 is started by writing a ‘1’ to bit TCTRL3[TEN]; bit TIE stays at 0.

The following example code matches the described setup:

Example 9

```
// turn on PIT
PIT_MCR = 0x00;

// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE;    // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN;   // start RTI

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE;        // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN;         // start Timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3for 1500000 cycles
PIT_TCTRL3 |= TEN;         // start Timer 3
```

44.5 Example Configuration for the Lifetime Timer

To configure the life timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a ‘0’ to field MCR[MDIS], then the LDVAL registers need to be set to the maximum value.

The following example code matches the described setup:

Example 10

```
// turn on PIT
PIT_CTRL = 0x00;

// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0;        // disable timer 2 interrupts
PIT_TCTRL1 |= CHN;       // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN;         // start timer 1

// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0for maximum counting period
PIT_TCTRL0 = TEN;          // start timer 0
```

To access the lifetime read first LTMR64H then LTMR64L.

Equation 29

current_uptime = PIT_LTMR64H << 32;

current_uptime = current_uptime + PIT_LTMR64L

The timer is a down counter.

45 Enhanced Modular IO Subsystem (eMIOS)

45.1 Introduction

45.1.1 Overview of the eMIOS module

The eMIOS provides functionality to generate or measure time events on up to 32 unified channels (UCs). Each UC provides a set of functional modes, 16-bit timer resolution, and provides a user interface that is consistent with previous eMIOS implementations.

45.1.2 Features of the eMIOS module

- Two eMIOS blocks with 32 channels
- eMIOS blocks can be synchronized from outside
- One global prescaler
- 16-bit data registers
- 16-bit wide counter buses
  - Counter buses B, C, D, and E can be driven by Unified Channel 0, 8, 16, and 24, respectively
  - Counter bus A is driven by the Unified Channel 23
  - Several channels have their own time base, alternative to the counter buses
  - Shared timebases through the counter buses
  - Synchronization among timebases
- Control and Status bits grouped in a single register
- Shadow FLAG register
- Flag outputs of channels 8-11 of eMIOSs are used to disable the outputs of other channels. They form the ODIS bits
- State of the UC can be frozen for debug purposes
- Motor control capability

45.1.3 Modes of operation

The Unified Channels can be configured to operate in the following modes:

- General purpose input/output
- Single Action Input Capture
- Single Action Output Compare
- Input Pulse Width Measurement
- Input Period Measurement
- Double Action Output Compare
- Modulus Counter
- Modulus Counter Buffered
- Output Pulse Width and Frequency Modulation Buffered
- Output Pulse Width Modulation Buffered
- Output Pulse Width Modulation with Trigger
- Center Aligned Output Pulse Width Modulation Buffered

These modes are described in *UC modes of operation*.

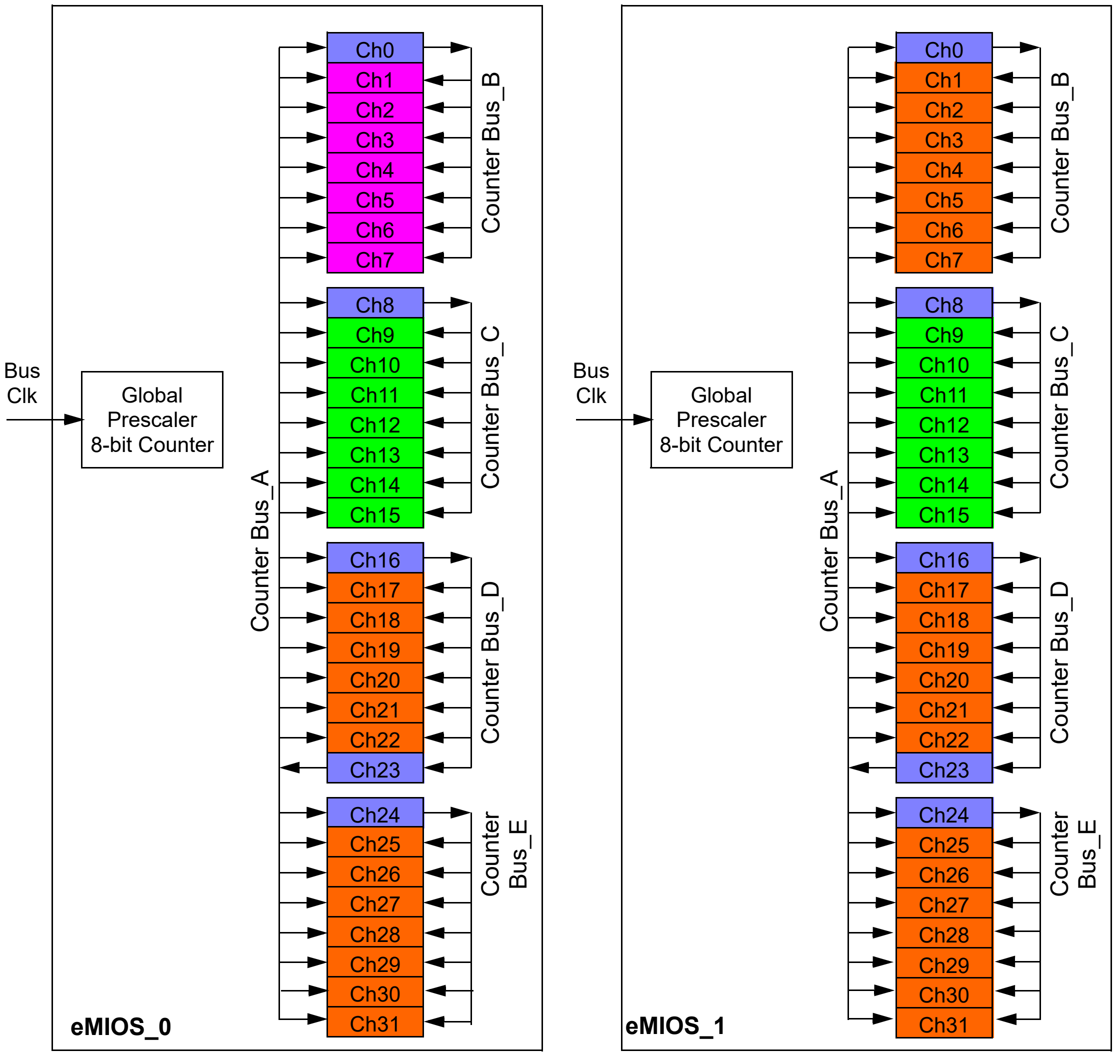
Each channel can have a specific set of modes implemented, according to device requirements.

If an unimplemented mode (reserved) is selected, the results are unpredictable such as writing a reserved value to MODE[25:31] in *eMIOS UC Control Register (EMIOSCn)*.

45.1.4 Channel implementation

Figure 468 shows the channel configuration of the eMIOS blocks.

Figure 468. Channel configuration



Channel Functionality

TYPE G	TYPE X	TYPE H	TYPE Y
• MCB • OPWMT • OPWMB • OPWFMB • OPWMCB • IPWM, IPM • DAOC • SAIC, SAOC • GPIO	• MC, MCB • OPWMT • OPWMB • OPWFMB • SAIC, SAOC • GPIO	• OPWMT • OPWMB • IPWM, IPM • DAOC • SAIC, SAOC • GPIO	• OPWMT • OPWMB • SAIC, SAOC • GPIO

Key

DAOC	Dual Action Output Compare	OPWMB	Buffered Output Pulse Width Modulation
GPIO	General Purpose Input Output	OPWMT	Buffered Output Pulse Width Modulation with Trigger
IPM	Input Period Measurement	OPWFMB	Buffered Output Pulse Width and Frequency Modulation
IPWM	Input Pulse Width Measurement	OPWMCB	Center Aligned Output PWM Buffered with Dead-Time
MC	Modulus Counter	SAIC	Single Action Input Capture
MCB	Buffered Modulus Counter	SAOC	Single Action Output Compare

45.1.4.1 Channel mode selection

Channel modes are selected using the mode selection bits MODE[25:31] in the *eMIOS UC Control Register (EMIOSCn)*. *Table 583* provides the specific mode selection settings for the eMIOS implementation on this device.

45.2 External signal description

For information on eMIOS external signals on this device, refer to the signal description chapter of the reference manual.

45.3 Memory map and register description

45.3.1 Memory maps

The overall address map organization is shown in *Table 567*.

45.3.1.1 Unified Channel memory map

Table 567. eMIOS memory map

Address offset	Register	Location
0x0000	eMIOS Module Configuration (EMIOSMCR)	*Section 45.3.2.1*
0x0004	eMIOS Global FLAG (EMIOSGFLAG)	*Section 45.3.2.2*
0x0008	eMIOS Output Update Disable (EMIOSOUDIS)	*Section 45.3.2.3*
0x000C	eMIOS Disable Channel (EMIOSUCDIS)	*Section 45.3.2.4*
0x0010–0x001F	Reserved	
0x0020–0x011F	Channel [0] to Channel [7]	—
0x0120–0x021F	Channel [8] to Channel [15]	—
0x0220–0x031F	Channel [16] to Channel [23]	—
0x0320–0x041F	Channel [24] to Channel [31]	—
0x0420–0x0FFF	Reserved	

Addresses of Unified Channel registers are specified as offsets from the channel's base address; otherwise the eMIOS base address is used as reference.

Table 568 describes the Unified Channel memory map.

Table 568. Unified Channel memory map

UC base address	Register	Location
0x0000	eMIOS UC A (EMIOSAn)	*Section 45.3.2.5*
0x0004	eMIOS UC B (EMIOSBn)	*Section 45.3.2.6*
0x0008	eMIOS UC Counter (EMIOSCNTn)	*Section 45.3.2.7*
0x000C	eMIOS UC Control (EMIOSCn)	*Section 45.3.2.8*
0x0010	eMIOS UC Status (EMIOSSn)	*Section 45.3.2.9*
0x0014	eMIOS UC Alternate A (EMIOSALTAn)	*Section 45.3.2.10*
0x0018–0x001F	Reserved	

45.3.2 Register description

All control registers are 32 bits wide. Data registers and counter registers are 16 bits wide.

45.3.2.1 eMIOS Module Configuration Register (EMIOSMCR)

The EMIOSMCR contains global control bits for the eMIOS block.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MDIS	FRZ	GTBE	0	GPREN	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPRE								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 469. eMIOS Module Configuration Register (EMIOSMCR)

Table 569. EMIOSMCR field descriptions

Field	Description
1 MDIS	Module Disable Puts the eMIOS in low power mode. The MDIS bit is used to stop the clock of the block, except the access to registers EMIOSMCR, EMIOSOUDIS and EMIOSUCDIS. 0 Clock is running 1 Enter low power mode
2 FRZ	Freeze Enables the eMIOS to freeze the registers of the Unified Channels when Debug Mode is requested at MCU level. Each Unified Channel should have FREN bit set in order to enter freeze state. While in Freeze state, the eMIOS continues to operate to allow the MCU access to the Unified Channels registers. The Unified Channel will remain frozen until the FRZ bit is written to ‘0’ or the MCU exits Debug mode or the Unified Channel FREN bit is cleared. 0 Exit freeze state 1 Stops Unified Channels operation when in Debug mode and the FREN bit is set in the EMIOSC*n* register
3 GTBE	Global Time Base Enable The GTBE bit is used to export a Global Time Base Enable from the module and provide a method to start time bases of several blocks simultaneously. 0 Global Time Base Enable Out signal negated 1 Global Time Base Enable Out signal asserted **Note:** The Global Time Base Enable Out signals of all eMIOS blocks are combined (ORed) together to generate one Global Time Base Enable Input signal that controls the internal counters of all eMIOS blocks. When asserted, internal counter are enable. When negated, internal counters are disabled.
5 GPREN	Global Prescaler Enable The GPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) and prescaler counter is cleared 1 Prescaler enabled
16:23 GPRE	Global Prescaler The GPRE bits select the clock divider value for the global prescaler, as shown in *Table 570*.

Table 570. Global prescaler clock divider

GPRE	Divide ratio
00000000	1
00000001	2
00000010	3
00000011	4
…	…
11111110	255
11111111	256

45.3.2.2 eMIOS Global FLAG (EMIOSGFLAG) Register

The EMIOSGFLAG is a read-only register that groups the flag bits (F[31:0]) from all unified channels.These bits are mirrors of the FLAG bits in each EMIOSS*n* register. This organization improves interrupt handling on simpler devices. Each bit relates to one channel.

Offset: 0x0004 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F31	F30	F29	F28	F27	F26	F25	F24	F23	F22	F21	F20	F19	F18	F17	F16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 470. eMIOS Global FLAG (EMIOSGFLAG) Register

Table 571. EMIOSGFLAG field descriptions

Field	Description
0:31 F[31:0]	Channel n Flag bit (F[n])

45.3.2.3 eMIOS Output Update Disable (EMIOSOUDIS) Register

Offset: 0x0008 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	OU31	OU30	OU29	OU28	OU27	OU26	OU25	OU24	OU23	OU22	OU21	OU20	OU19	OU18	OU17	OU16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	OU15	OU14	OU13	OU12	OU11	OU10	OU9	OU8	OU7	OU6	OU5	OU4	OU3	OU2	OU1	OU0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 471. eMIOS Output Update Disable (EMIOSOUDIS) Register

Table 572. EMIOSOUDIS field descriptions

Field	Description
0:31 OU[31:0]	Channel n Output Update Disable bit (OU[n]) When running MC, MCB or an output mode, values are written to registers A2 and B2. OU*n* bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel. 0 Transfer enabled. Depending on the operation mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. 1 Transfers disabled

45.3.2.4 eMIOS Disable Channel (EMIOSUCDIS) Register

Offset: 0x000C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CHDIS31	CHDIS30	CHDIS29	CHDIS28	CHDIS27	CHDIS26	CHDIS25	CHDIS24	CHDIS23	CHDIS22	CHDIS21	CHDIS20	CHDIS19	CHDIS18	CHDIS17	CHDIS16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CHDIS15	CHDIS14	CHDIS13	CHDIS12	CHDIS11	CHDIS10	CHDIS9	CHDIS8	CHDIS7	CHDIS6	CHDIS5	CHDIS4	CHDIS3	CHDIS2	CHDIS1	CHDIS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 472. eMIOS Enable Channel (EMIOSUCDIS) Register

Table 573. EMIOSUCDIS field descriptions

Field	Description
0:31 CHDIS[31:0]	Enable Channel n bit (CHDIS[n]) The CHDIS*n* bit is used to disable each of the channels by stopping its respective clock. 0 Channel *n* enabled 1 Channel *n* disabled

45.3.2.5 eMIOS UC A Register (EMIOSA*n*)

Depending on the mode of operation, internal registers A1 or A2, used for matches and captures, can be assigned to the EMIOSA*n* register. Both A1 and A2 are cleared by reset. *Table 576* summarizes the EMIOSA*n* writing and reading accesses for all operation modes. For more information, refer to *UC modes of operation*.

Offset: 0x0020 + n * 0x20 (n = 0 to 31) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	A															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 473. eMIOS UC A Register (EMIOSA*n*)

Table 574. eMIOS UC A Register (EMIOSA*n*) field descriptions

Field	Description
16:31 A	A1 or A2 depending on the mode of operation

45.3.2.6 eMIOS UC B Register (EMIOSB*n*)

Depending on the mode of operation, internal registers B1 or B2 can be assigned to the EMIOSB*n* register. Both B1 and B2 are cleared by reset. *Table 576* summarizes the EMIOSB*n* writing and reading accesses for all operation modes. For more information, refer to *UC modes of operation*.

Depending on the channel configuration, it may have EMIOSB register or not. This means that, if at least one mode that requires the register is implemented, then the register is present; otherwise it is absent.

Offset: 0x0024 + n * 0x20 (n = 0 to 31) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	B															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 474. eMIOS UC B Register (EMIOSB*n*)

Table 575. eMIOS UC B Register (EMIOSB*n*) field descriptions

Field	Description
16:31 B	B1 or B2 depending on the mode of operation

Table 576. EMIOSA*n*, EMIOSB*n* and EMIOSALTA*n* values assignment

Operation mode	Register access					
	write	read	write	read	alt write	alt read
GPIO	A1, A2	A1	B1,B2	B1	A2	A2
SAIC(1)	—	A2	B2	B2	—	—
SAOC(1)	A2	A1	B2	B2	—	—
IPWM	—	A2	—	B1	—	—
IPM	—	A2	—	B1	—	—
DAOC	A2	A1	B2	B1	—	—
MC(1)	A2	A1	B2	B2	—	—
OPWMT	A1	A1	B2	B1	A2	A2
MCB(1)	A2	A1	B2	B2	—	—
OPWFMB	A2	A1	B2	B1	—	—
OPWMCB	A2	A1	B2	B1	—	—
OPWMB	A2	A1	B2	B1	—	—

1. In these modes, the register EMIOSB*n* is not used, but B2 can be accessed.

45.3.2.7 eMIOS UC Counter Register (EMIOSCNT*n*)

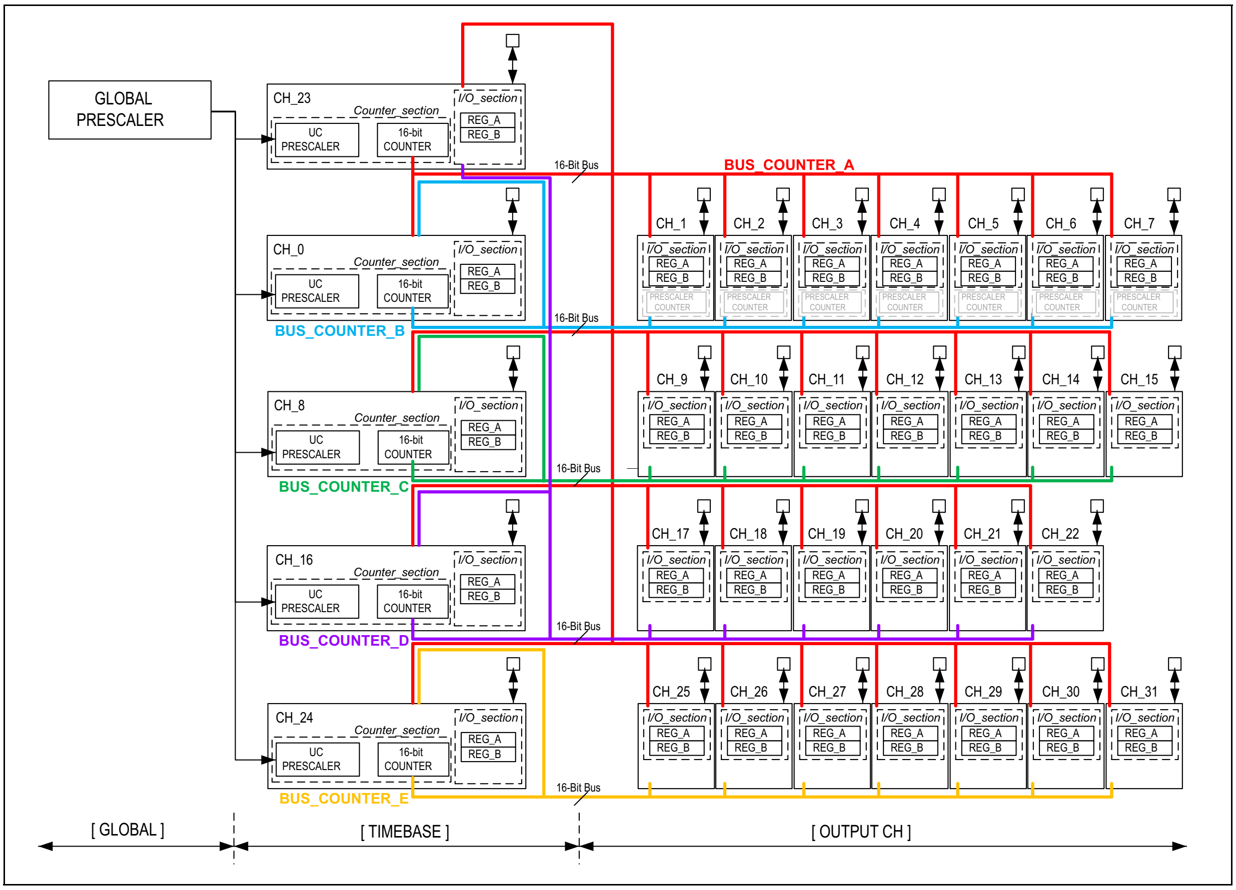
The EMIOSCNT*n* register contains the value of the internal counter. When GPIO mode is selected or the channel is frozen, the EMIOSCNT*n* register is read/write. For all other modes, the EMIOSCNT*n* is a read-only register. When entering some operation modes, this register is automatically cleared (refer to *UC modes of operation* for details).

Depending on the channel configuration it may have an internal counter or not. It means that if at least one mode that requires the counter is implemented, then the counter is present; otherwise it is absent.

Channels of type X and G have the internal counter enabled, so their timebase can be selected by channel's BSL[21:22] = 11:eMIOS_A - channels 0 to 8, 16, 23 and 24, eMIOS_B = channels 0, 8, 16, 23 and 24. Other channels from the above list do not have internal counters.

Figure 475 shows the bus counter choices for the unified channels.

Figure 475. eMIOS bus counters



Note: EMIOS_0: counter_section is implemented in CH_0/8/16/23/24 and CH_1 to 7

Note: EMIOS_1: counter_section is implemented in CH_0/8/16/23/24

Offset: 0x0028 + n * 0x20 (n = 0 to 31) Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W(1)																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C															
W(1)																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In GPIO mode or Freeze action, this register is writable.

Figure 476. eMIOS UC Counter Register (EMIOSCNT*n*)

Table 577. eMIOS UC Counter Register (EMIOSCNT*n*) field descriptions

Field	Description
16:31 C	Internal counter value

45.3.2.8 eMIOS UC Control Register (EMIOSC*n*)

The Control register gathers bits reflecting the status of the UC input/output signals and the overflow condition of the internal counter, as well as several read/write control bits.

Offset: 0x002C + n * 0x20 (n = 0 to 31) Access: Read/Write

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FREN	ODIS	ODISSL		UCPRE		UCPREN	DMA	0	IF				FCK	FEN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BSL		EDSEL	EDPOL	MODE						
W			FORCMA	FORCMB												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 477. eMIOS UC Control Register (EMIOSC*n*)

Table 578. EMIOSC*n* field descriptions

Field	Description
0 FREN	Freeze Enable bit The FREN bit, if set and validated by FRZ bit in EMIOSMCR register allows the channel to enter freeze state, freezing all registers values when in debug mode and allowing the MCU to perform debug functions. 0 Normal operation 1 Freeze UC registers values
1 ODIS	Output Disable bit The ODIS bit allows disabling the output pin when running any of the output modes with the exception of GPIO mode. 1 If the selected Output Disable Input signal is asserted, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, i.e., it continues to produce FLAG and matches. When the selected Output Disable Input signal is negated, the output pin operates normally. 0 The output pin operates normally.
2:3 ODISSL	Output Disable Select bits The ODISSL[1:0] bits select one of the four output disable input signals, as shown in *Table 579*.

Table 578. EMIOSC*n* field descriptions (continued)

Field	Description
4:5 UCPRE	Prescaler bits The UCPRE bits select the clock divider value for the internal prescaler of Unified Channel, as shown in *Table 580*.
6 UCPREN	Prescaler Enable bit The UCPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) 1 Prescaler enabled
7 DMA	Direct Memory Access bit The DMA bit selects if the FLAG signal will be used as an interrupt request, as a DMA request or as a CTU trigger. The choice between a DMA request or a CTU trigger is determined by the value of bit TM in the register CTU_EVTCFGRx (refer to the CTU chapter of the reference manual). 0 Flag/overrun assigned to interrupt request 1 Flag/overrun assigned to DMA request or CTU trigger
9:12 IF	Input Filter The IF field controls the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in *Table 581*. For output modes, these bits have no meaning.
13 FCK	Filter Clock select bit The FCK bit selects the clock source for the programmable input filter. 0 Prescaled clock 1 Main clock
14 FEN	FLAG Enable bit The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal or a CTU trigger signal (The type of signal to be generated is defined by the DMA bit). 0 Disable (FLAG does not generate an interrupt request or DMA request or a CTU trigger) 1 Enable (FLAG will generate an interrupt request or DMA request or a CTU trigger)
18 FORCMA	Force Match A bit For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A, otherwise it has no effect. 0 Has no effect 1 Force a match at comparator A **Note:** For input modes, the FORCMA bit is not used and writing to it has no effect.
19 FORCMB	Force Match B bit For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B, otherwise it has no effect. 0 Has no effect 1 Force a match at comparator B **Note:** For input modes, the FORCMB bit is not used and writing to it has no effect.

Table 578. EMIOSC*n* field descriptions (continued)

Field	Description
21:22 BSL	Bus Select The BSL field is used to select either one of the counter buses or the internal counter (if present) to be used by the Unified Channel. Refer to *Table 582* for details.
23 EDSEL	Edge Selection bit For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit. When not shown in the mode of operation description, this bit has no effect. 0 [All input modes]: single edge triggering defined by the EDPOL bit; [GPIO input mode]: a FLAG is generated as defined by the EDPOL bit; [SAOC mode]: the output flip-flop is toggled at each match. 1 [All input modes]: both edges triggering; [GPIO input mode]: no FLAG is generated; [SAOC mode]: the output flip-flop is toggled at each match.
24 EDPOL	Edge Polarity bit For input modes, the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG. When not shown in the mode of operation description, this bit has no effect. 0 [All input modes]: trigger on a falling edge; [All output modes]: a match on comparator A clears the output flip-flop, while a match on comparator B sets it 1 [All input modes]: trigger on a rising edge; [All output modes]: a match on comparator A sets the output flip-flop, while a match on comparator B clears it
25:31 MODE	Mode selection The MODE field selects the mode of operation of the Unified Channel, as shown in *Table 583*. **Note:** If a reserved value is written to mode the results are unpredictable.

Table 579. UC ODISSL selection

ODISSL	Input signal
00	Output Disable Input 0
01	Output Disable Input 1
10	Output Disable Input 2
11	Output Disable Input 3

Table 580. UC internal prescaler clock divider

UCPRE	Divide ratio
00	1
01	2
10	3
11	4

Table 581. UC input filter bits

IF[1]	Minimum input pulse width [FLT_CLK periods]
0000	Bypassed[2]
0001	02
0010	04
0100	08
1000	16
all others	Reserved

1. Filter latency is 3 main clock periods.
2. The input signal is synchronized before arriving to the digital filter.

Table 582. UC BSL bits

BSL	Selected bus
00	All channels: counter bus[A]
01	Channels 0 to 7: counter bus[B] Channels 8 to 15: counter bus[C] Channels 16 to 23: counter bus[D] Channels 24 to 31: counter bus[E]
10	Reserved
11	UC internal counter (use only if counter present)

Table 583. Channel mode selection

MODE[1]	Mode of operation
0000000	General purpose Input/Output mode (input)
0000001	General purpose Input/Output mode (output)
0000010	Single Action Input Capture
0000011	Single Action Output Compare
0000100	Input Pulse Width Measurement
0000101	Input Period Measurement
0000110	Double Action Output Compare (with FLAG set on B match)
0000111	Double Action Output Compare (with FLAG set on both matches)
0001000 – 0001111	Reserved
001000b	Modulus Counter (Up counter with clear on match start)
001001b	Modulus Counter (Up counter with clear on match end)
00101bb	Modulus Counter (Up/Down counter)
0011000 – 0100101	Reserved
0100110	Output Pulse Width Modulation with Trigger
0100111 – 1001111	Reserved

Table 583. Channel mode selection (continued)

MODE[1]	Mode of operation
101000b	Modulus Counter Buffered (Up counter)
101001b	Reserved
10101bb	Modulus Counter Buffered (Up/Down counter)
10110b0	Output Pulse Width and Frequency Modulation Buffered
10110b1	Reserved
10111b0	Center Aligned Output Pulse Width Modulation Buffered (with trail edge dead-time)
10111b1	Center Aligned Output Pulse Width Modulation Buffered (with lead edge dead-time)
11000b0	Output Pulse Width Modulation Buffered
1100001 – 1111111	Reserved

1. b = adjust parameters for the mode of operation. Refer to *UC modes of operation* for details.

45.3.2.9 eMIOS UC Status Register (EMIOSS*n*)

Offset: 0x0030 + n * 0x20 (n = 0 to 31) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OVFL	0	0	0	0	0	0	0	0	0	0	0	0	UCIN	UCOUT	FLAG
W	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 478. eMIOS UC Status Register (EMIOSS*n*)

Table 584. EMIOSS*n* field descriptions

Field	Description
0 OVR	Overrun bit The OVR bit indicates that FLAG generation occurred when the FLAG bit was already set. 0 Overrun has not occurred 1 Overrun has occurred
16 OVFL	Overflow bit The OVFL bit indicates that an overflow has occurred in the internal counter. OVFL must be cleared by software writing a 1 to the OVFLC bit. 0 No overflow 1 An overflow had occurred

Table 584. EMIOSS*n* field descriptions (continued)

Field	Description
29 UCIN	Unified Channel Input pin bit The UCIN bit reflects the input pin state after being filtered and synchronized.
30 UCOUT	Unified Channel Output pin bit The UCOUT bit reflects the output pin state.
31 FLAG	FLAG bit The FLAG bit is set when an input capture or a match event in the comparators occurred. 0 FLAG cleared 1 FLAG set event has occurred **Note:** When DMA bit is set, the FLAG bit can be cleared by the DMA controller or the CTU.

45.3.2.10 eMIOS UC Alternate A Register (EMIOSALTA*n*)

The EMIOSALTA*n* register provides an alternate register to access A2 channel registers in restricted modes (GPIO, OPWMT) only. If EMIOSA*n* register is used along with EMIOSALTA*n*, both A1 and A2 registers can be accessed in these modes. *Figure 576* summarizes the EMIOSALTA*n* writing and reading accesses for all operation modes. Refer to *General purpose Input/Output (GPIO) mode*, *Output Pulse Width Modulation with Trigger (OPWMT) mode* for a more detailed description of the use of EMIOSALTA*n* register.

Offset: 0x0034 + n * 0x20 (n = 0 to 31) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ALTA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 479. eMIOS UC Alternate A register (EMIOSALTA*n*)

Table 585. eMIOS UC Alternate A register (EMIOSALTA*n*) field descriptions

Field	Description
16:31 ALTA	Alternate address to access the A2 channel registers in restricted mode

45.4 Functional description

The four types of channels of the eMIOS (types X, Y, G and H) can operate in the modes as listed in *Figure 468*. The eMIOS provides independently operating unified channels (UC) that can be configured and accessed by a host MCU. Up to three time bases(o) can be shared by the channels through five counter buses[p] and each unified channel can generate its own time base[q]. The eMIOS block is reset at positive edge of the clock (synchronous reset). All registers are cleared on reset.

45.4.1 Unified Channel (UC)

Each Unified Channel consists of:

- Counter bus selector, which selects the time base to be used by the channel for all timing functions
- A programmable clock prescaler
- Two double buffered data registers A and B that allow up to two input capture and/or output compare events to occur before software intervention is needed
- Two comparators (equal only) A and B, which compare the selected counter bus with the value in the data registers
- Internal counter, which can be used as a local time base or to count input events
- Programmable input filter, which ensures that only valid pin transitions are received by channel
- Programmable input edge detector, which detects the rising, falling or either edges
- An output flip-flop, which holds the logic level to be applied to the output pin
- eMIOS Status and Control register

45.4.1.1 UC modes of operation

The mode of operation of the Unified Channel is determined by the mode select bits MODE[25:31] in the *eMIOS UC Control Register (EMIOSCn)* (refer to *Figure 477* for details).

As the internal counter EMIOSCNT*n* continues to run in all modes (except for GPIO mode), it is possible to use this local counter as a timebase for other UCs if it is not used for the current local mode.

In order to provide smooth waveform generation, a few specific modes allow A and B registers to be changed on the fly. These modes are suffixed with a “B” (MCB, OPWFMB, OPWMB and OPWMCB), meaning that in these modes A and B registers are double buffered.

45.4.1.1.1 General purpose Input/Output (GPIO) mode

In GPIO mode, all input capture and output compare functions of the UC are disabled, the internal counter (EMIOSCNT*n* register) is cleared and disabled. All control bits remain

---

o. Time bases can be supplied by:
i) channel 23 to all unified channels
ii) channel 0 to channels 0 to 7, by channel 8 to channels 8 to 15, by channel 16 to channels 16 to 23, by channel 24 to channels 24 to 31
iii) channel's internal counter when available.

p. Internal eMIOS architecture has one global counter bus A and four local counter buses B, C, D, and E, that distribute the time bases described in footnote *o.* (i) and (ii).

q. Channels of type X and G have the internal counter enabled, so their timebase can be selected by channel's BSL[21:22]=11: eMIOS_A - channels 0 to 8, 16, 23 and 24 eMIOS_B = channels 0, 8, 16, 23 and 24.

accessible. In order to prepare the UC for a new operation mode, writing to registers EMIOSA*n* or EMIOSB*n* stores the same value in registers A1/A2 or B1/B2, respectively. Writing to register EMIOSALTA*n* stores a value only in register A2.

MODE[31] bit selects between input (MODE[31] = 0) and output (MODE[31] = 1) modes.

It is required that when changing MODE[25:31], the application software goes to GPIO mode first in order to reset the UC's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results or the FLAGs being set incorrectly.

In GPIO input mode (MODE[25:31] = 0000000), the FLAG generation is determined according to EDPOL and EDSEL bits and the input pin status can be determined by reading the UCIN bit.

In GPIO output mode (MODE[25:31] = 0000001), the Unified Channel is used as a single output port pin and the value of the EDPOL bit is permanently transferred to the output flip-flop.

45.4.1.1.2 Single Action Input Capture (SAIC) mode

In SAIC mode (MODE[25:31] = 0000010), when a triggering event occurs on the input pin, the value on the selected time base is captured into register A2. The FLAG bit is set along with the capture event to indicate that an input capture has occurred. Register EMIOSA*n* returns the value of register A2. As soon as the SAIC mode is entered coming out from GPIO mode the channel is ready to capture events. The events are captured as soon as they occur thus reading register A always returns the value of the latest captured event. Subsequent captures are enabled with no need of further reads from EMIOSA*n* register. The FLAG is set at any time a new event is captured.

The input capture is triggered by a rising, falling or either edges in the input pin, as configured by EDPOL and EDSEL bits in EMIOSC*n* register.

Figure 480 and *Figure 481* show how the Unified Channel can be used for input capture.

Figure 480. Single action input capture with rising edge triggering example

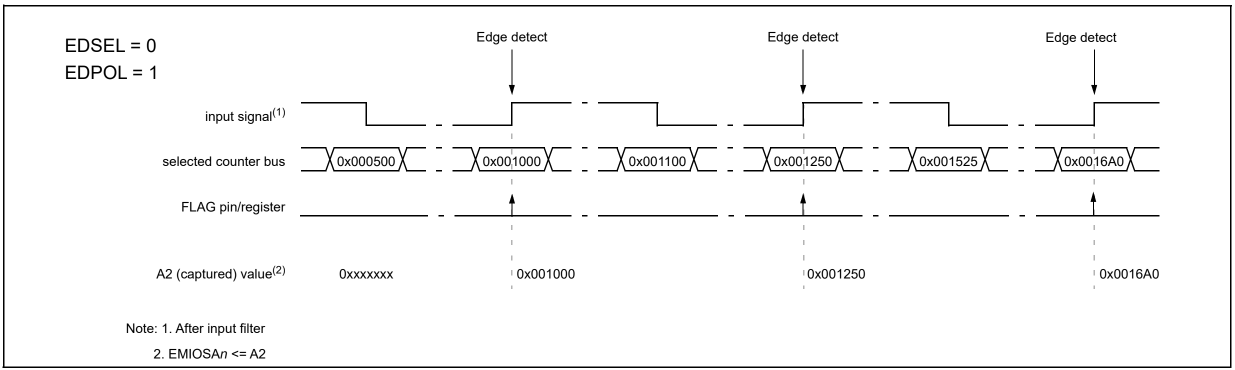
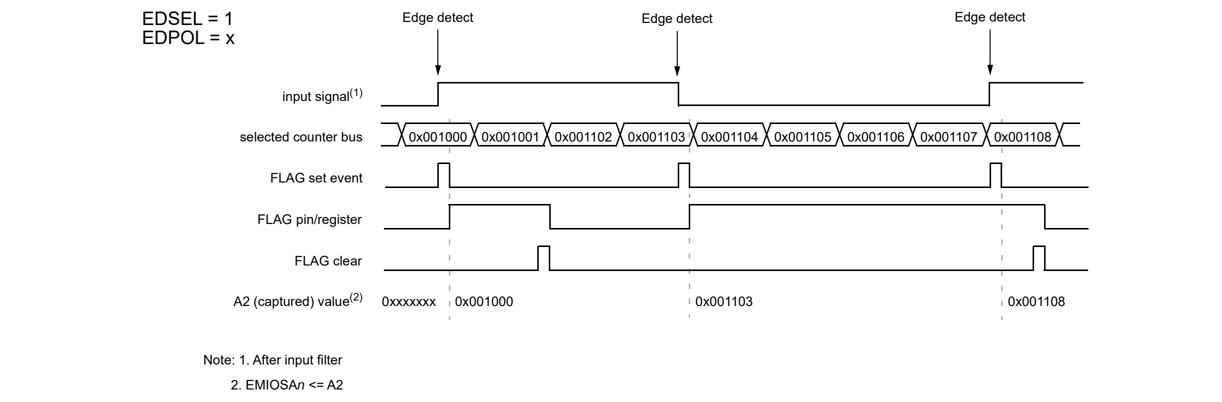


Figure 481. Single action input capture with both edges triggering example



45.4.1.1.3 Single Action Output Compare (SAOC) mode

In SAOC mode (MODE[25:31] = 0000011) a match value is loaded in register A2 and then immediately transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit selects whether the output flip-flop is toggled or the value in EDPOL is transferred to it. Along with the match the FLAG bit is set to indicate that the output compare match has occurred. Writing to register EMIOSA*n* stores the value in register A2 and reading to register EMIOSA*n* returns the value of register A1.

An output compare match can be simulated in software by setting the FORCMA bit in EMIOSC*n* register. In this case, the FLAG bit is not set.

When SAOC mode is entered coming out from GPIO mode the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC*n* register.

Counter bus can be either internal or external and is selected through bits BSL[21:22].

Figure 482 and *Figure 483* show how the Unified Channel can be used to perform a single output compare with EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively. Note that once in SAOC mode the matches are enabled thus the desired match value on register A1 must be written before the mode is entered. A1 register can be updated at any time thus modifying the match value which will reflect in the output signal generated by the channel. Subsequent matches are enabled with no need of further writes to EMIOSA*n* register. The FLAG is set at the same time a match occurs (refer to *Figure 484*).

Note: The channel internal counter in SAOC mode is free-running. It starts counting as soon as the SAOC mode is entered.

Figure 482. SAOC example with EDPOL value being transferred to the output flip-flop

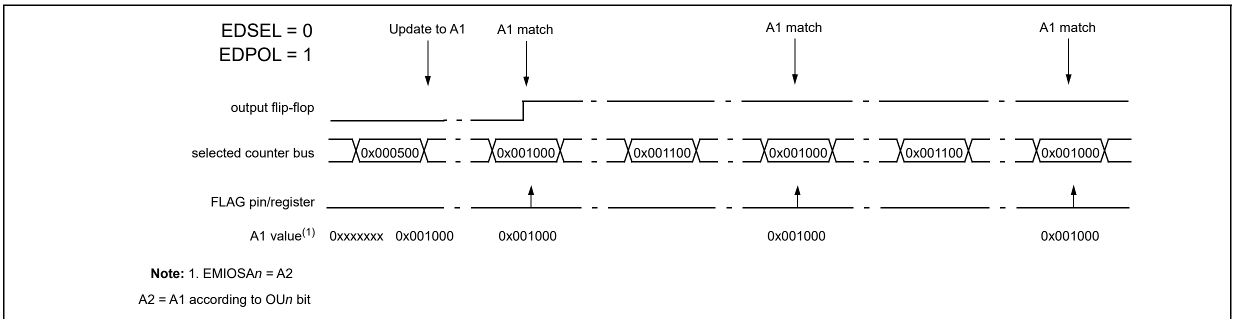


Figure 483. SAOC example toggling the output flip-flop

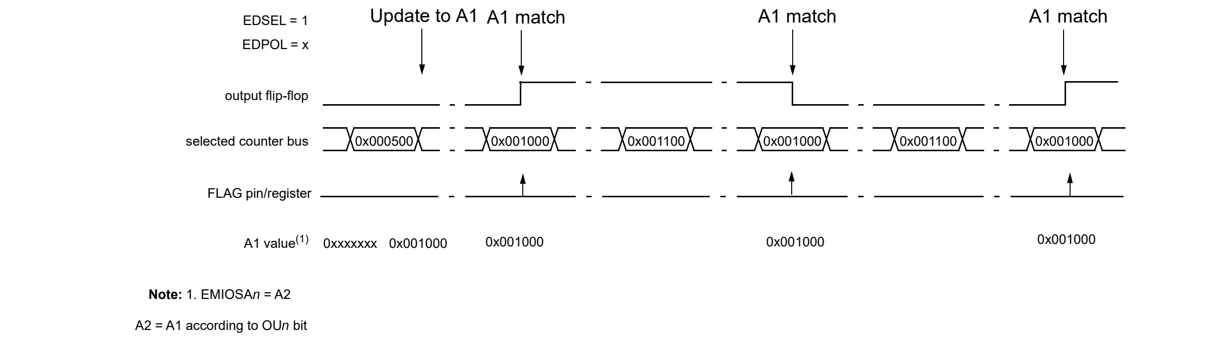
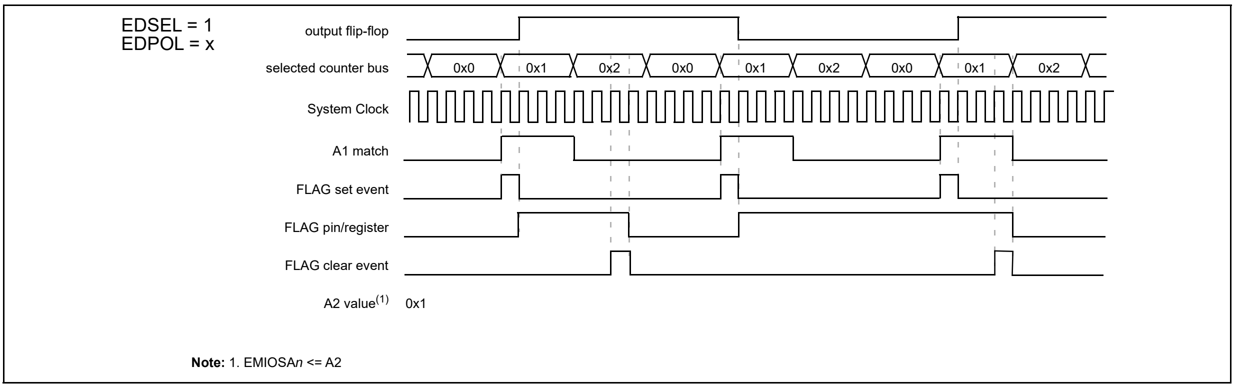


Figure 484. SAOC example with flag behavior



45.4.1.1.4 Input Pulse Width Measurement (IPWM) Mode

The IPWM mode (MODE[25:31] = 0000100) allows the measurement of the width of a positive or negative pulse by capturing the leading edge on register B1 and the trailing edge on register A2. Successive captures are done on consecutive edges of opposite polarity. The leading edge sensitivity (that is, pulse polarity) is selected by EDPOL bit in the EMIOSC*n* register. Registers EMIOSA*n* and EMIOSB*n* return the values in register A2 and B1, respectively.

The capture function of register A2 remains disabled until the first leading edge triggers the first input capture on register B2. When this leading edge is detected, the count value of the selected time base is latched into register B2; the FLAG bit is not set. When the trailing edge is detected, the count value of the selected time base is latched into register A2 and, at the same time, the FLAG bit is set and the content of register B2 is transferred to register B1 and to register A1.

If subsequent input capture events occur while the corresponding FLAG bit is set, registers A2, B1 and A1 will be updated with the latest captured values and the FLAG will remain set. Registers EMIOSA*n* and EMIOSB*n* return the value in registers A2 and B1, respectively.

In order to guarantee coherent access, reading EMIOSA*n* forces B1 be updated with the content of register A1. At the same time transfers between B2 and B1 are disabled until the next read of EMIOSB*n* register. Reading EMIOSB*n* register forces B1 be updated with A1 register content and re-enables transfers from B2 to B1, to take effect at the next trailing edge capture. Transfers from B2 to A1 are not blocked at any time.

The input pulse width is calculated by subtracting the value in B1 from A2.

Figure 485 shows how the Unified Channel can be used for input pulse width measurement.

Figure 485. Input pulse width measurement example

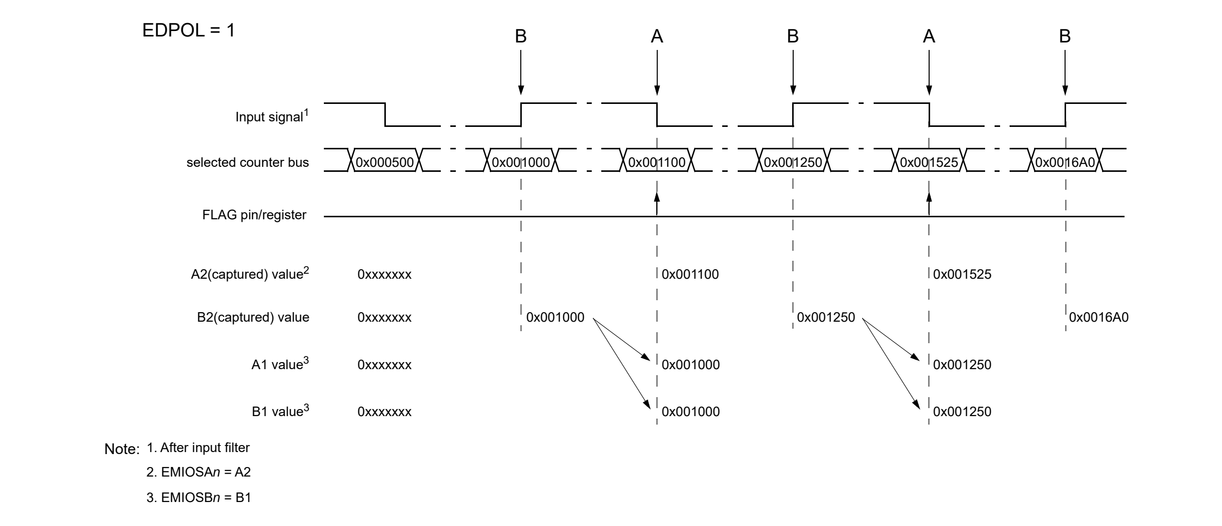
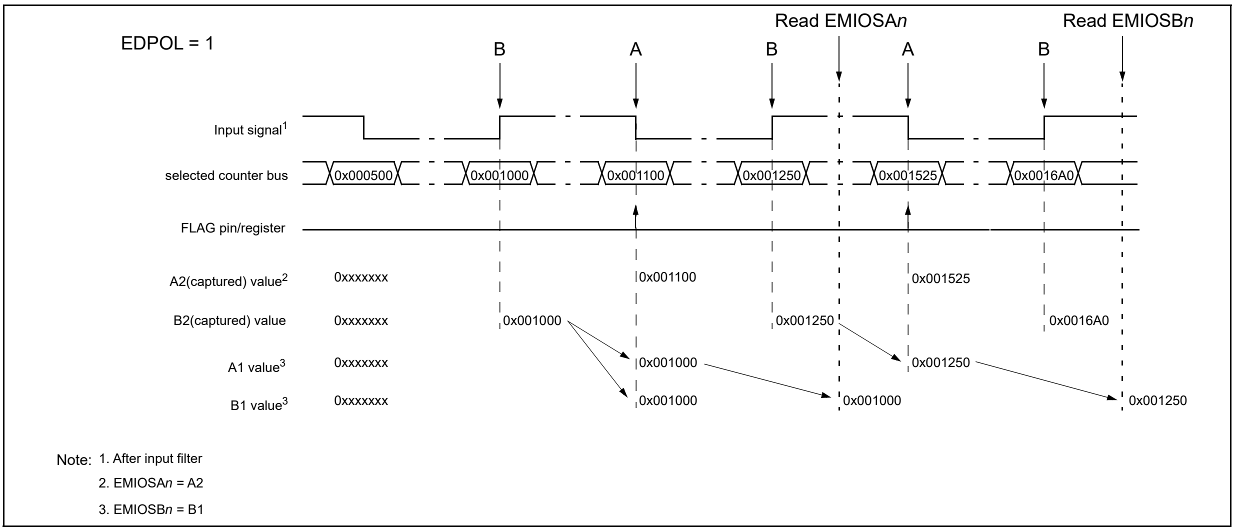


Figure 486 shows the A1 and B1 updates when EMIOSA*n* and EMIOSB*n* register reads occur. Note that A1 register has always coherent data related to A2 register. Note also that when EMIOSA*n* read is performed B1 register is loaded with A1 register content. This guarantees that the data in register B1 has always the coherent data related to the last EMIOSA*n* read. The B1 register updates remain locked until EMIOSB*n* read occurs. If EMIOSA*n* read is performed B1 is updated with A1 register content, even if B1 update is locked by a previous EMIOSA*n* read operation.

Figure 486. B1 and A1 updates at EMIOSA*n* and EMIOSB*n* reads



Reading EMIOSA*n* followed by EMIOSB*n* always provides coherent data. If not coherent data is required for any reason, the sequence of reads should be inverted, therefore EMIOSB*n* should be read prior to EMIOSA*n* register. Note that even in this case B1 register updates will be blocked after EMIOSA*n* read, thus a second EMIOSB*n* is required in order to release B1 register updates.

45.4.1.1.5 Input Period Measurement (IPM) mode

The IPM mode (MODE[25:31] = 0000101) allows the measurement of the period of an input signal by capturing two consecutive rising edges or two consecutive falling edges. Successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the EMIOSC*n* register.

Note: The input signal must have at least four system clock cycles period in order to be properly captured by the synchronization logic at the channel input even if the input filter is in by-pass mode.

When the first edge of selected polarity is detected, the selected time base is latched into the registers A2 and B2, and the data previously held in register B2 is transferred to register B1. On this first capture the FLAG line is not set, and the values in registers B1 is meaningless. On the second and subsequent captures, the FLAG line is set and data in register B2 is transferred to register B1.

When the second edge of the same polarity is detected, the counter bus value is latched into registers A2 and B2, the data previously held in register B2 is transferred to data register B1 and to register A1. The FLAG bit is set to indicate the start and end points of a complete period have been captured. This sequence of events is repeated for each subsequent capture. Registers EMIOSA*n* and EMIOSB*n* return the values in register A2 and B1, respectively.

In order to allow coherent data, reading EMIOSA*n* forces A1 content be transferred to B1 register and disables transfers between B2 and B1. These transfers are disabled until the next read of the EMIOSB*n* register. Reading EMIOSB*n* register forces A1 content to be transferred to B1 and re-enables transfers from B2 to B1, to take effect at the next edge capture.

The input pulse period is calculated by subtracting the value in B1 from A2.

Figure 487 shows how the Unified Channel can be used for input period measurement.

Figure 487. Input period measurement example

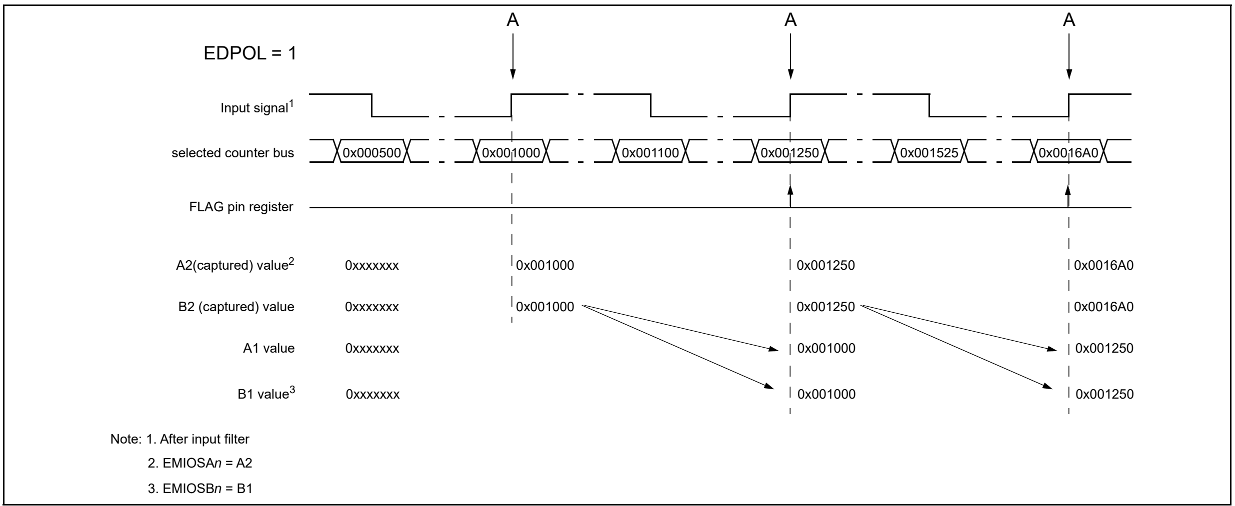
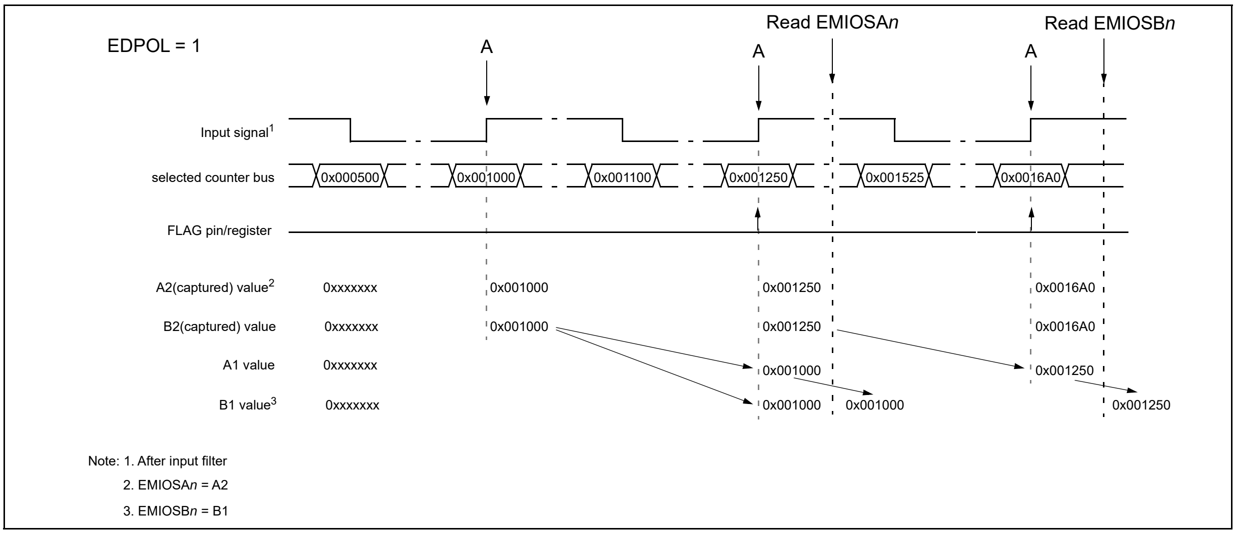


Figure 488 describes the A1 and B1 register updates when EMIOSA*n* and EMIOSB*n* read operations are performed. When EMIOSA*n* read occurs the content of A1 is transferred to B1 thus providing coherent data in A2 and B1 registers. Transfers from B2 to B1 are then blocked until EMIOSB*n* is read. After EMIOSB*n* is read, register A1 content is transferred to

register B1 and the transfers from B2 to B1 are re-enabled to occur at the transfer edges, which is the leading edge in the *Figure 488* example.

Figure 488. A1 and B1 updates at EMIOSA*n* and EMIOSB*n* reads



45.4.1.1.6 Double Action Output Compare (DAOC) mode

In the DAOC mode the leading and trailing edges of the variable pulse width output are generated by matches occurring on comparators A and B. There is no restriction concerning the order in which A and B matches occur.

When the DAOC mode is entered, coming out from GPIO mode both comparators are disabled and the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC*n* register.

Data written to A2 and B2 are transferred to A1 and B1, respectively, on the next system clock cycle if bit OU*n* of the EMIOSOUDIS register is cleared (refer to *Figure 491*). The transfer is blocked if bit OU*n* is set. Comparator A is enabled only after the transfer to A1 register occurs and is disabled on the next A match. Comparator B is enabled only after the transfer to B1 register occurs and is disabled on the next B match. Comparators A and B are enabled and disabled independently.

The output flip-flop is set to the value of EDPOL when a match occurs on comparator A and to the complement of EDPOL when a match occurs on comparator B.

MODE[31] controls if the FLAG is set on both matches (MODE[25:31] = 0000111) or just on the B match (MODE[25:31] = 0000110). FLAG bit assertion depends on comparator enabling.

If subsequent enabled output compares occur on registers A1 and B1, pulses will continue to be generated, regardless of the state of the FLAG bit.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a comparison event in comparator A or B, respectively. Note that the FLAG bit is not affected by these forced operations.

Note: If both registers (A1 and B1) are loaded with the same value, the B match prevails concerning the output pin state (output flip-flop is set to the complement of EDPOL), the FLAG bit is set and both comparators are disabled.

Figure 489 and *Figure 490* show how the Unified Channel can be used to generate a single output pulse with FLAG bit being set on the second match or on both matches, respectively.

Figure 489. Double action output compare with FLAG set on the second match

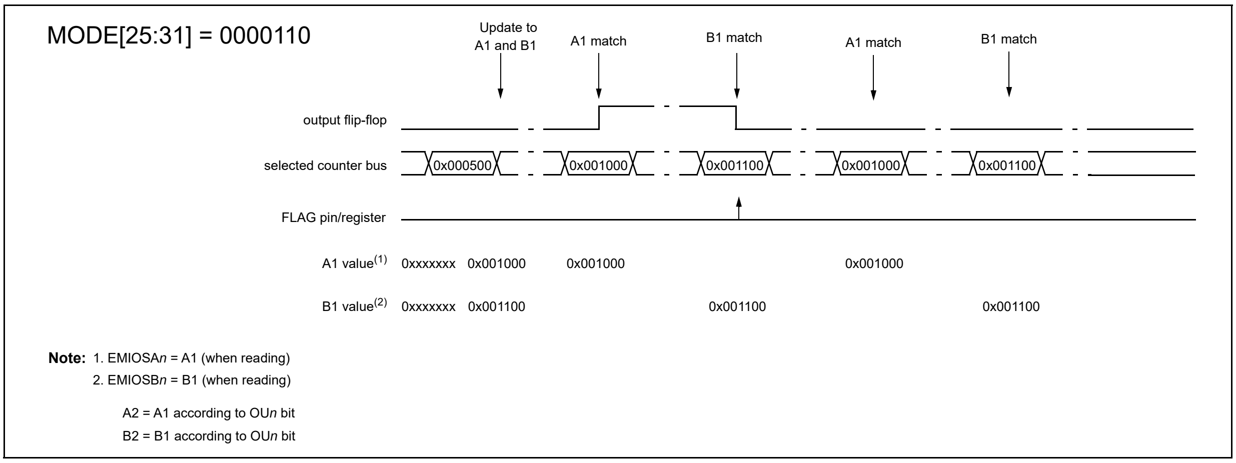


Figure 490. Double action output compare with FLAG set on both matches

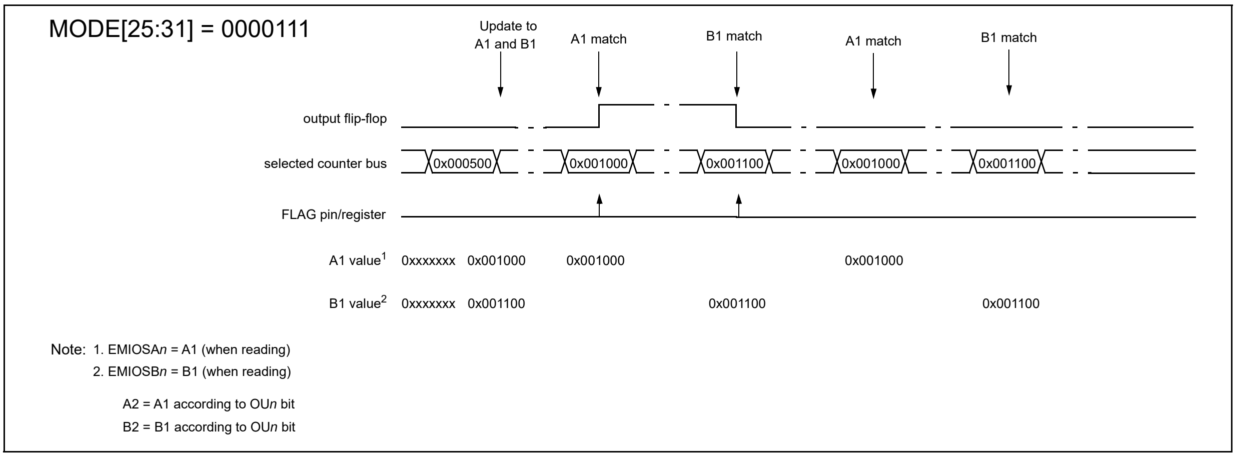
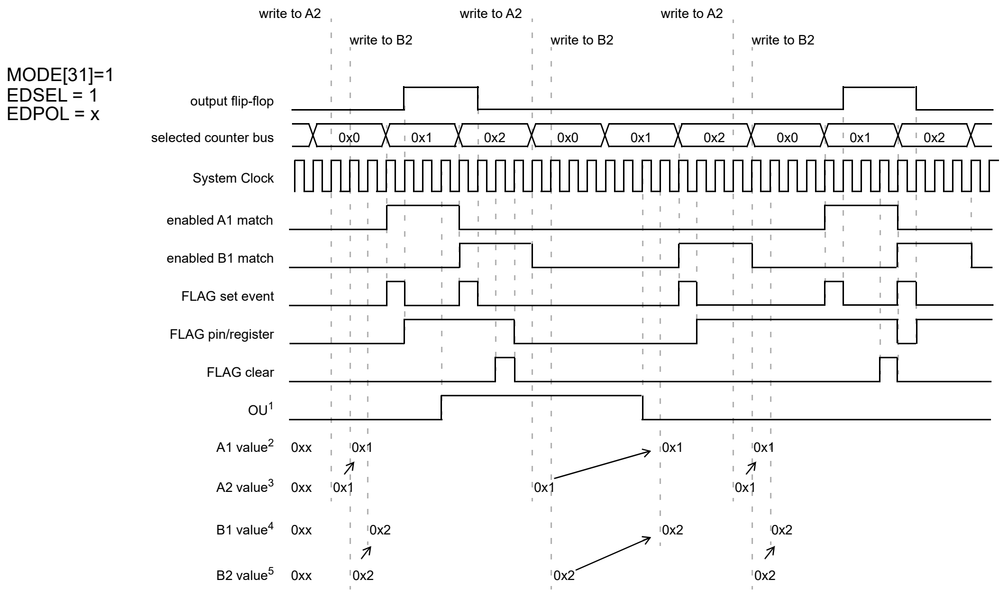


Figure 491. DAOC with transfer disabling example



Note: 1. OU*n* bit of EMIOSOUDIS register
2. EMIOSA*n* = A1 (when reading)
3. EMIOSA*n* = A2 (when writing)
4. EMIOSB*n* = B1 (when reading)
5. EMIOSB*n* = B2 (when writing)

45.4.1.1.7 Modulus Counter (MC) mode

The MC mode can be used to provide a time base for a counter bus or as a general purpose timer.

Bit MODE[31] selects internal or external clock source when cleared or set, respectively. When external clock is selected, the input signal pin is used as the source and the triggering polarity edge is selected by the EDPOL and EDSEL in the EMIOSC*n* register.

The internal counter counts up from the current value until it matches the value in register A1. Register B1 is cleared and is not accessible to the MCU. Bit MODE[29] selects up mode or up/down mode, when cleared or set, respectively.

When in up count mode, a match between the internal counter and register A1 sets the FLAG and clears the internal counter. The timing of those events varies according to the MC mode setup as follows:

- Internal counter clearing on match start (MODE[25:31] = 001000b)
  - External clock is selected if MODE[31] is set. In this case the internal counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. Note that by having the internal counter cleared as soon as the match occurs and incremented at the next input event a shorter zero count is generated. Refer to *Figure 514* and *Figure 515*.
  - Internal clock source is selected if MODE[31] is cleared. In this case the counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. At the next prescaler tick after the match the internal

counter remains at zero and only resumes counting on the following tick. Refer to *Figure 514* and *Figure 516*.

- Internal counter clearing on match end (MODE[25:31] = 001001b)
  – External clock is selected if MODE[31] is set. In this case the internal counter clears when the match signal is asserted and the input event occurs. The channel FLAG is set at the same time the counter is cleared. Refer to *Figure 514* and *Figure 517*.
  – Internal clock source is selected if MODE[31] is cleared. In this case the internal counter clears when the match signal is asserted and the prescaler tick occurs. The channel FLAG is set at the same time the counter is cleared. Refer to *Figure 514* and *Figure 517*.

Note: If the internal clock source is selected and the prescaler of the internal counter is set to '1', the MC mode behaves the same way even in Clear on Match Start or Clear on Match End submodes.

When in up/down count mode (MODE[25:31] = 00101bb), a match between the internal counter and register A1 sets the FLAG and changes the counter direction from increment to decrement. A match between register B1 and the internal counter changes the counter direction from decrement to increment and sets the FLAG only if MODE[30] bit is set.

Only values different than 0x0 must be written at A register. Loading 0x0 leads to unpredictable results.

Updates on A register or counter in MC mode may cause loss of match in the current cycle if the transfer occurs near the match. In this case, the counter may rollover and resume operation in the next cycle.

Register B2 has no effect in MC mode. Nevertheless, register B2 can be accessed for reads and writes by addressing EMIOSB.

Figure 492 and *Figure 493* show how the Unified Channel can be used as modulus counter in up mode and up/down mode, respectively.

Figure 492. Modulus Counter Up mode example

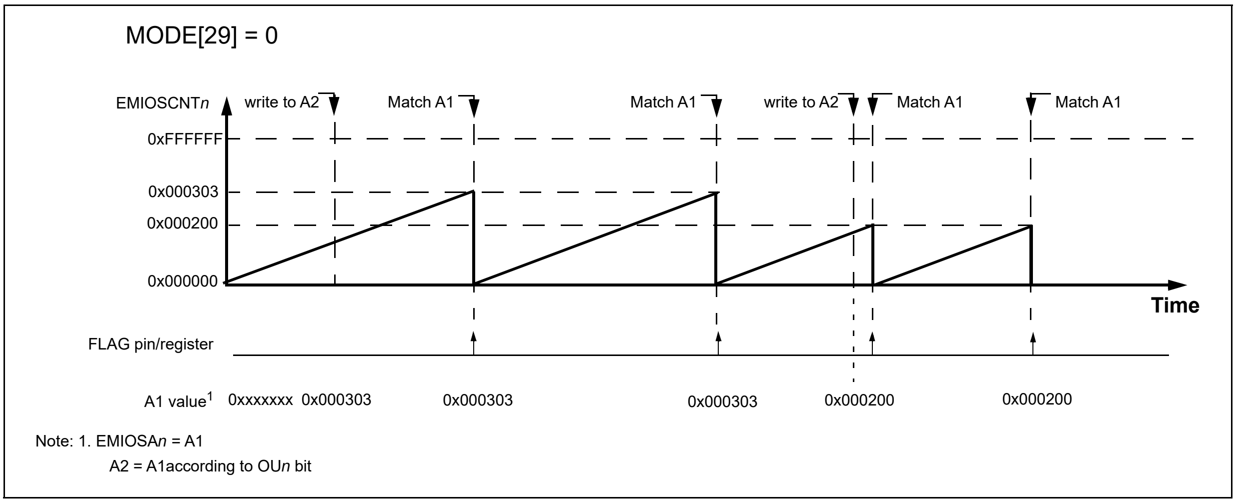
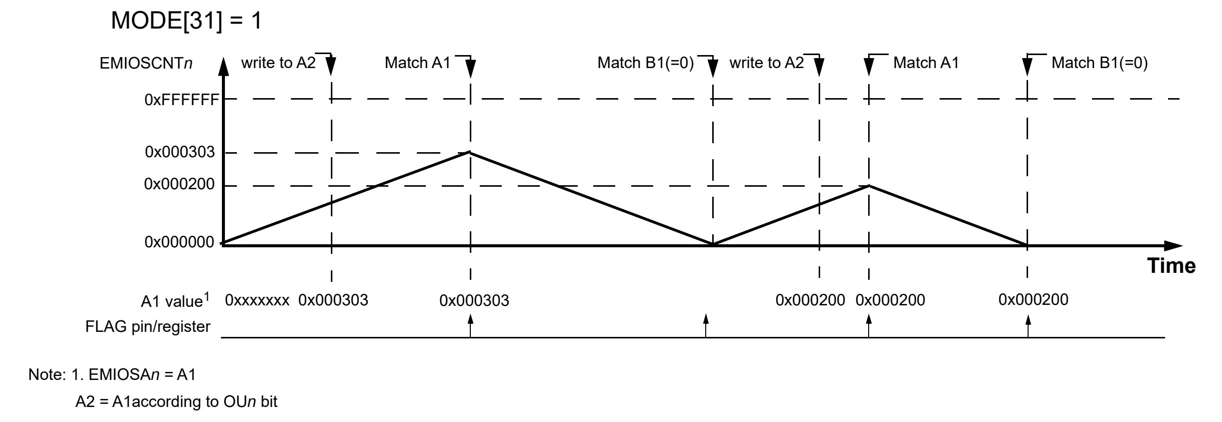


Figure 493. Modulus Counter Up/Down mode example



45.4.1.1.8 Modulus Counter Buffered (MCB) mode

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double buffered thus allowing smooth transitions between cycles when changing A2 register value on the fly. A1 register is updated at the cycle boundary, which is defined as when the internal counter transitions to 0x1.

The internal counter values operates within a range from 0x1 up to register A1 value. If when entering MCB mode coming out from GPIO mode the internal counter value is not within that range then the A match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xFFFF for a 16-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal MCB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to A1 register value range when the MCB mode is entered.

Bit MODE[31] selects internal clock source if cleared or external if set. When external clock is selected the input channel pin is used as the channel clock source. The active edge of this clock is defined by EDPOL and EDSEL bits in the EMIOSC*n* channel register.

When entering in MCB mode, if up counter is selected by MODE[29] = 0 (MODE[25:31] = 101000b), the internal counter starts counting from its current value to up direction until A1 match occurs. The internal counter is set to 0x1 when its value matches A1 value and a clock tick occurs (either prescaled clock or input pin event).

If up/down counter is selected by setting MODE[29] = 1, the counter changes direction at A1 match and counts down until it reaches the value 0x1. After it has reached 0x1 it is set to count in up direction again. B1 register is used to generate a match in order to set the internal counter in up-count direction if up/down mode is selected. Register B1 cannot be changed while this mode is selected.

Note that differently from the MC mode, the MCB mode counts between 0x1 and A1 register value. Only values greater than 0x1 must be written at A1 register. Loading values other than those leads to unpredictable results. The counter cycle period is equal to A1 value in up counter mode. If in up/down counter mode the period is defined by the expression: (2*A1)-2.

Figure 494 describes the counter cycle for several A1 values. Register A1 is loaded with A2 register value at the cycle boundary. Thus any value written to A2 register within cycle **n** will be updated to A1 at the next cycle boundary and therefore will be used on cycle **n+1**. The cycle boundary between cycle **n** and cycle **n+1** is defined as when the internal counter transitions from A1 value in cycle **n** to 0x1 in cycle **n+1**. Note that the FLAG is generated at the cycle boundary and has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

Figure 494. Modulus Counter Buffered (MCB) Up Count mode

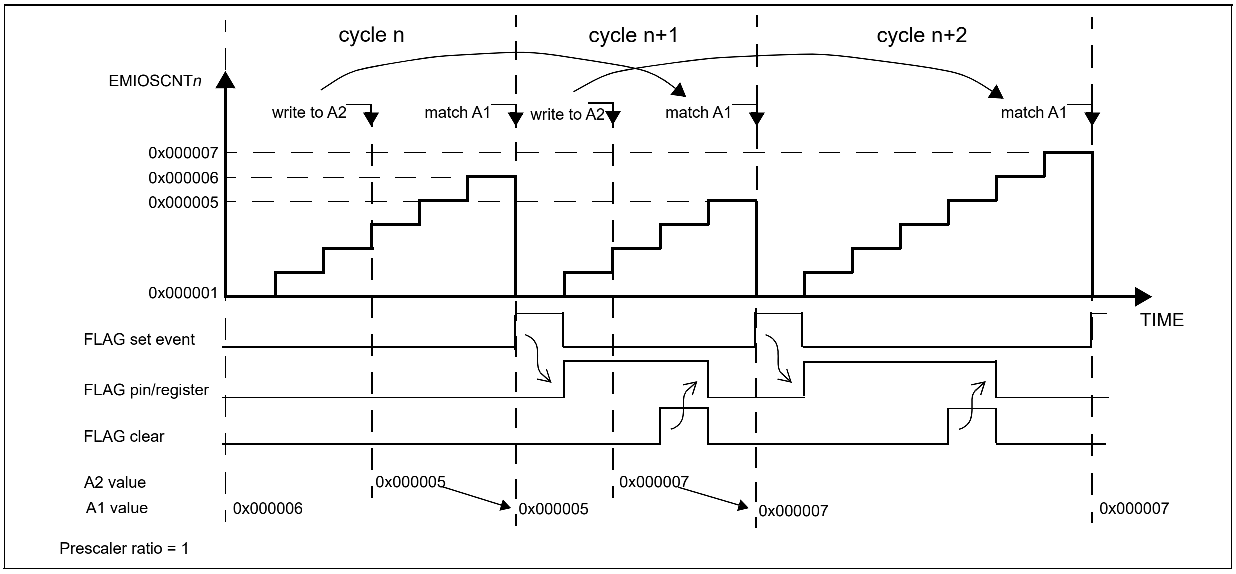


Figure 495 describes the MCB in up/down counter mode (MODE[25:31] = 10101bb). A1 register is updated at the cycle boundary. If A2 is written in cycle **n**, this new value will be used in cycle **n+1** for A1 match. Flags are generated only at A1 match start if MODE[30] is 0. If MODE[30] is set to 1 flags are also generated at the cycle boundary.

Figure 495. Modulus Counter Buffered (MCB) Up/Down mode

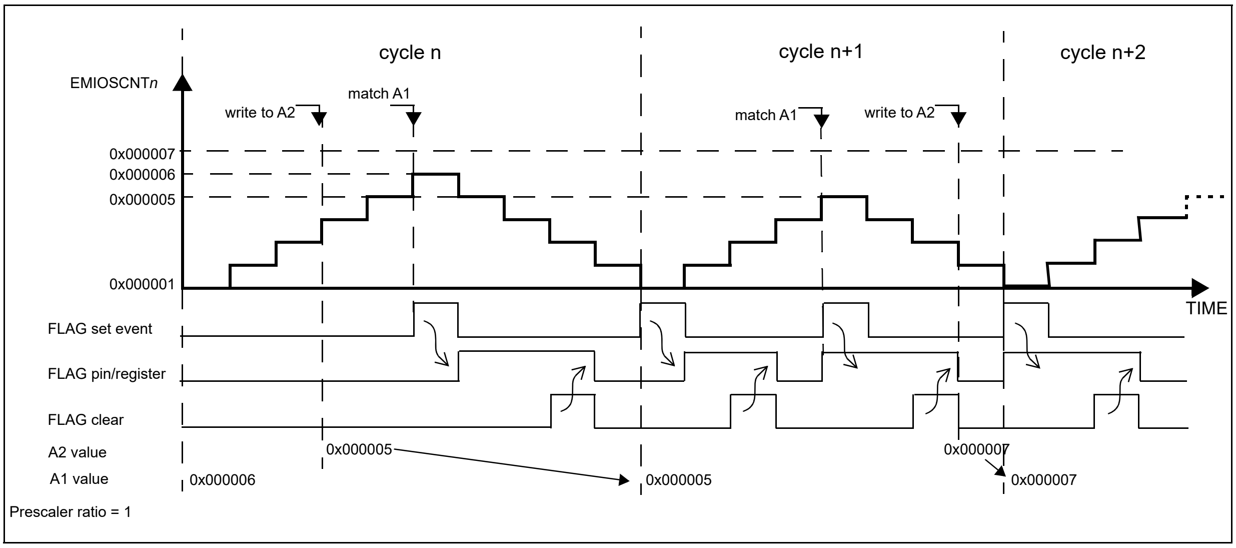


Figure 496 describes in more detail the A1 register update process in up counter mode. The A1 load signal is generated at the last system clock period of a counter cycle. Thus, A1 is updated with A2 value at the same time that the counter (EMIOSCNT*n*) is loaded with 0x1. The load signal pulse has the duration of one system clock period. If A2 is written within cycle **n** its value is available at A1 at the first clock of cycle **n+1** and the new value is used

for match at cycle **n+1**. The update disable bits OU*n* of EMIOSOUDIS register can be used to control the update of this register, thus allowing to delay the A1 register update for synchronization purposes.

Figure 496. MCB Mode A1 Register Update in Up Counter mode

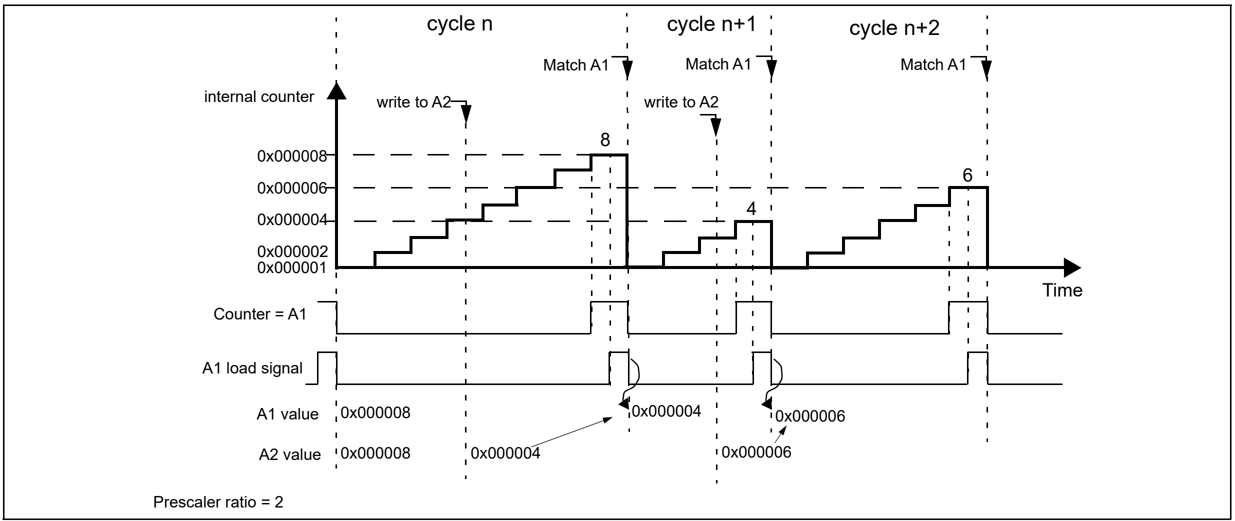
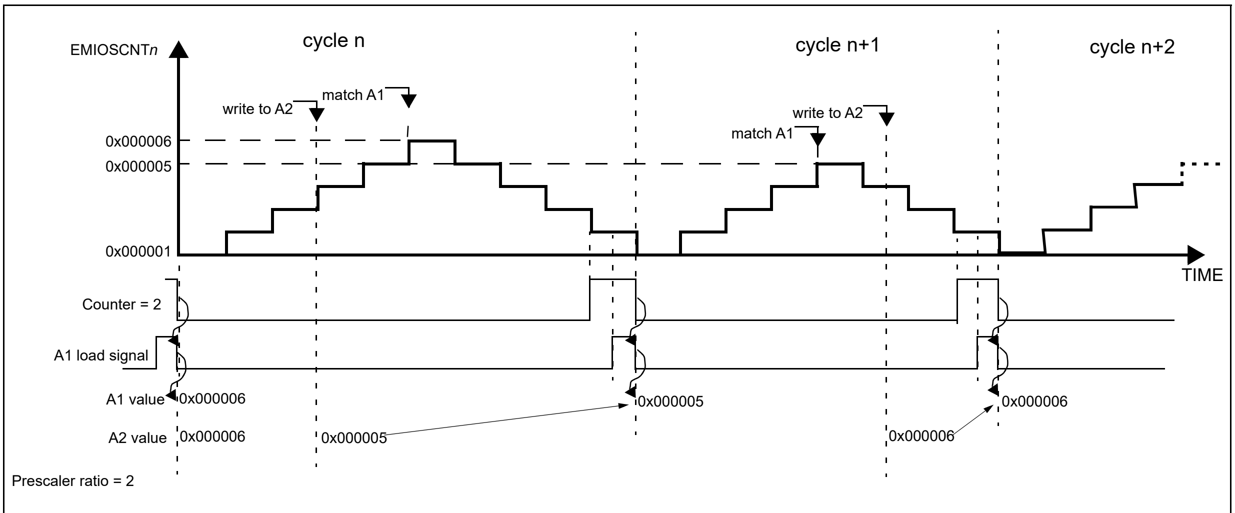


Figure 497 describes the A1 register update in up/down counter mode. Note that A2 can be written at any time within cycle **n** in order to be used in cycle **n+1**. Thus A1 receives this new value at the next cycle boundary. Note that the update disable bits OU*n* of EMIOSOUDIS register can be used to disable the update of A1 register.

Figure 497. MCB Mode A1 Register Update in Up/Down Counter mode



45.4.1.1.9 Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode

This mode (MODE[25:31] = 10110b0) provides waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base when this mode is selected. A1 register indicates the duty cycle and B1 register the frequency. Both A1 and B1 registers are double buffered to allow smooth signal generation when changing the registers values on the fly. 0% and 100% duty cycles are supported.

At OPWFMB mode entry the output flip-flop is set to the value of the EDPOL bit in the EMIOSC*n* register.

If when entering OPWFMB mode coming out from GPIO mode the internal counter value is not within that range then the B match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xFFFF for a 16-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal OPWFMB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to B1 register value range when the OPWFMB mode is entered.

When a match on comparator A occurs the output register is set to the value of EDPOL. When a match on comparator B occurs the output register is set to the complement of EDPOL. B1 match also causes the internal counter to transition to 0x1, thus restarting the counter cycle.

Only values greater than 0x1 are allowed to be written to B1 register. Loading values other than those leads to unpredictable results. If you want to configure the module for OPWFMB mode, ensure that the B1 register is modified before the mode is set.

Figure 498 describes the operation of the OPWFMB mode regarding output pin transitions and A1/B1 registers match events. Note that the output pin transition occurs when the A1 or B1 match signal is deasserted which is indicated by the A1 match negedge detection signal. If register A1 is set to 0x4 the output pin transitions 4 counter periods after the cycle had started, plus one system clock cycle. Note that in the example shown in *Figure 498* the internal counter prescaler has a ratio of two.

Figure 498. OPWFMB A1 and B1 match to Output Register Delay

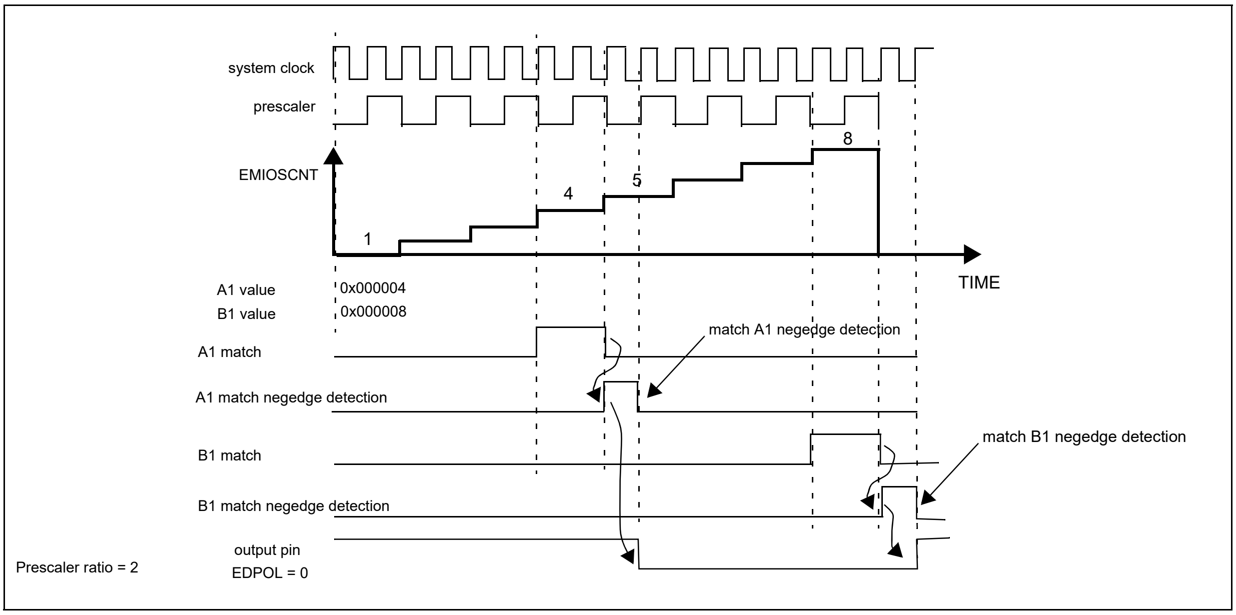


Figure 499 describes the generated output signal if A1 is set to 0x0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1 = 0x1 with the difference that in this case, the posedge of the match signal is used to trigger the output pin transition instead of the negedge used when A1 = 0x1. Note that A1 posedge match signal from cycle **n+1** occurs at the same time as B1 negedge match signal from cycle **n**. This allows to use the A1 posedge match to mask the B1 negedge match when they occur at the same time. The result is that no transition occurs on the output flip-flop and a 0% duty cycle is generated.

Figure 499. OPWFMB Mode with A1 = 0 (0% duty cycle)

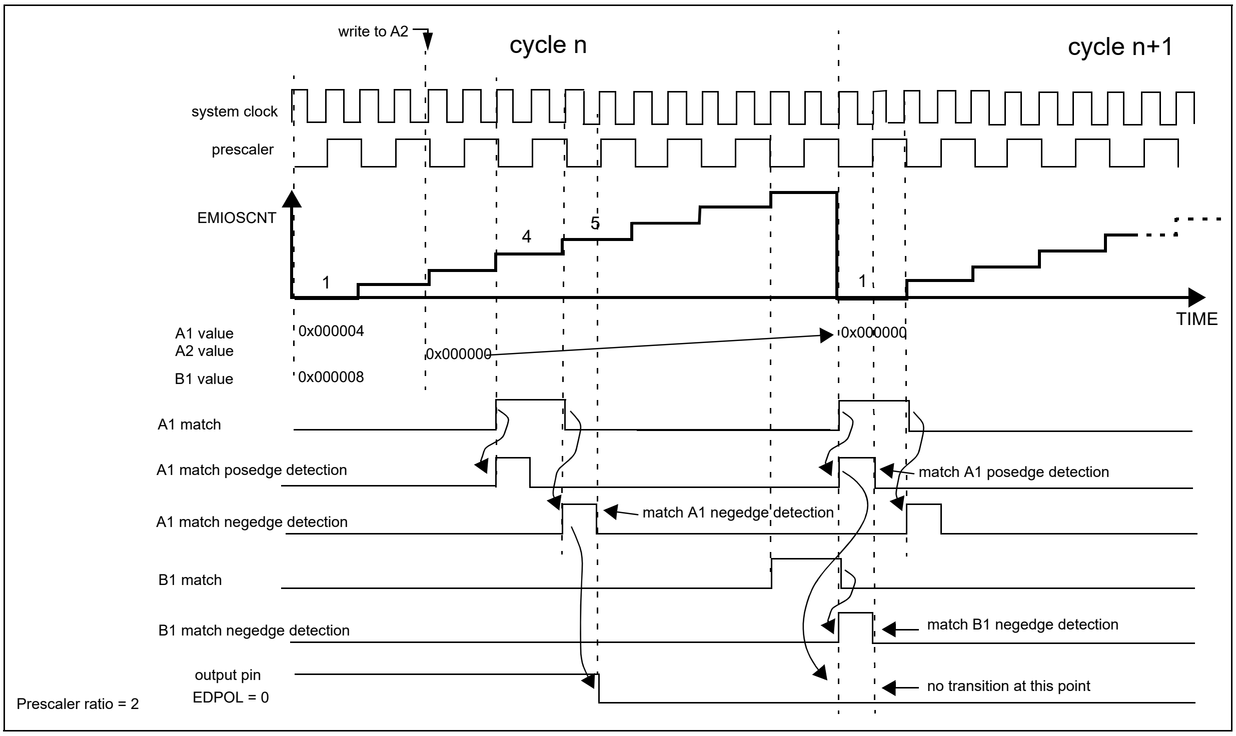
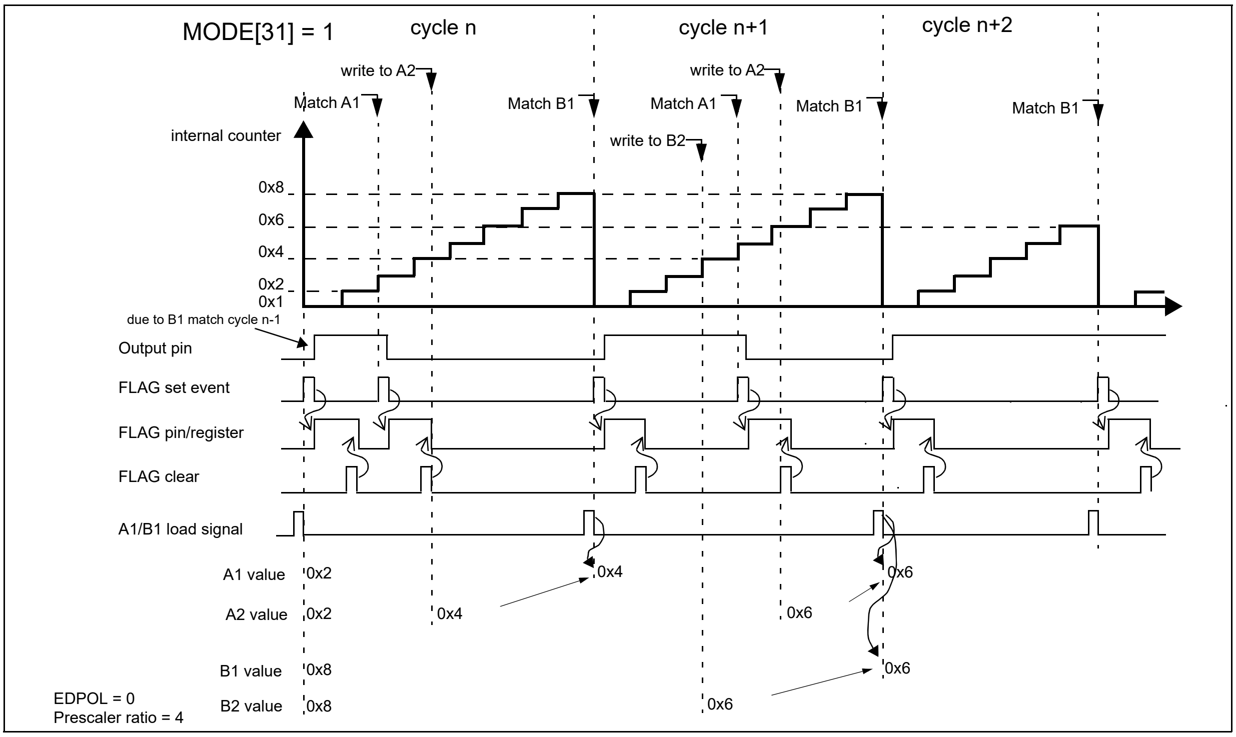


Figure 500 describes the timing for the A1 and B1 registers load. The A1 and B1 load use the same signal which is generated at the last system clock period of a counter cycle. Thus, A1 and B1 are updated respectively with A2 and B2 values at the same time that the counter (EMIOSCNT*n*) is loaded with 0x1. This event is defined as the cycle boundary. The load signal pulse has the duration of one system clock period. If A2 and B2 are written within cycle **n** their values are available at A1 and B1, respectively, at the first clock of cycle **n+1** and the new values are used for matches at cycle **n+1**. The update disable bits OU*n* of EMIOSOUDIS register can be used to control the update of these registers, thus allowing to delay the A1 and B1 registers update for synchronization purposes.

In *Figure 500* it is assumed that both the channel and global prescalers are set to 0x1 (each divide ratio is two), meaning that the channel internal counter transitions at every four system clock cycles. FLAGs can be generated only on B1 matches when MODE[30] is cleared, or on both A1 and B1 matches when MODE[30] is set. Since B1 flag occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle **n** were loaded to A1 or B1, respectively, thus generating matches in cycle **n+1**. Note that the FLAG has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

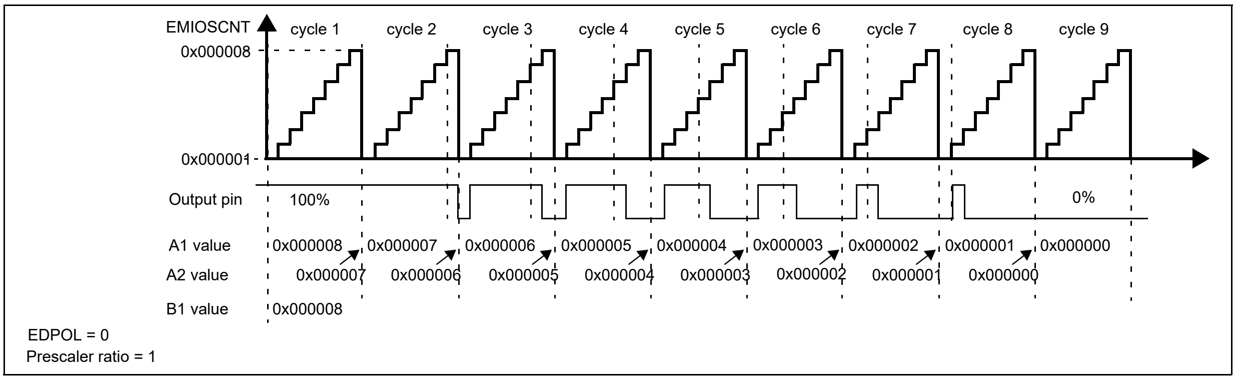
Figure 500. OPWFMB A1 and B1 registers update and flags



The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B respectively. Similarly to a B1 match FORCMB sets the internal counter to 0x1. The FLAG bit is not set by the FORCMA or FORCMB bits being asserted.

Figure 501 describes the generation of 100% and 0% duty cycle signals. It is assumed EDPOL = 0 and the resultant prescaler value is 1. Initially A1 = 0x8 and B1 = 0x8. In this case, B1 match has precedence over A1 match, thus the output flip-flop is set to the complement of EDPOL bit. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater than or equal to B1.

Figure 501. OPWFMB mode from 100% to 0% duty cycle



A 0% duty cycle signal is generated if A1 = 0x0 as shown in *Figure 501* cycle 9. In this case B1 = 0x8 match from cycle 8 occurs at the same time as the A1 = 0x0 match from cycle 9. Refer to *Figure 499* for a description of the A1 and B1 match generation. In this case A1 match has precedence over B1 match and the output signal transitions to EDPOL.

45.4.1.1.10 Center Aligned Output PWM Buffered with Dead-Time (OPWMCB) mode

This operation mode generates a center aligned PWM with dead time insertion to the leading (MODE[25:31] = 10111b1) or trailing edge (MODE[25:31] = 10111b0). A1 and B1 registers are double buffered to allow smooth output signal generation when changing A2 or B2 registers values on the fly.

Bits BSL[21:22] select the time base. The time base selected for a channel configured to OPWMCB mode should be a channel configured to MCB Up/Down mode, as shown in *Figure 495*. It is recommended to start the MCB channel time base after the OPWMCB mode is entered in order to avoid missing A matches at the very first duty cycle.

Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base.

Register B1 contains the dead time value and is compared against the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty cycle is equal to the sum of register A1 and register B1. Bit Mode[31] selects between trailing and leading dead time insertion, respectively.

Note: The internal counter runs in the internal prescaler ratio, while the selected time base may be running in a different prescaler ratio.

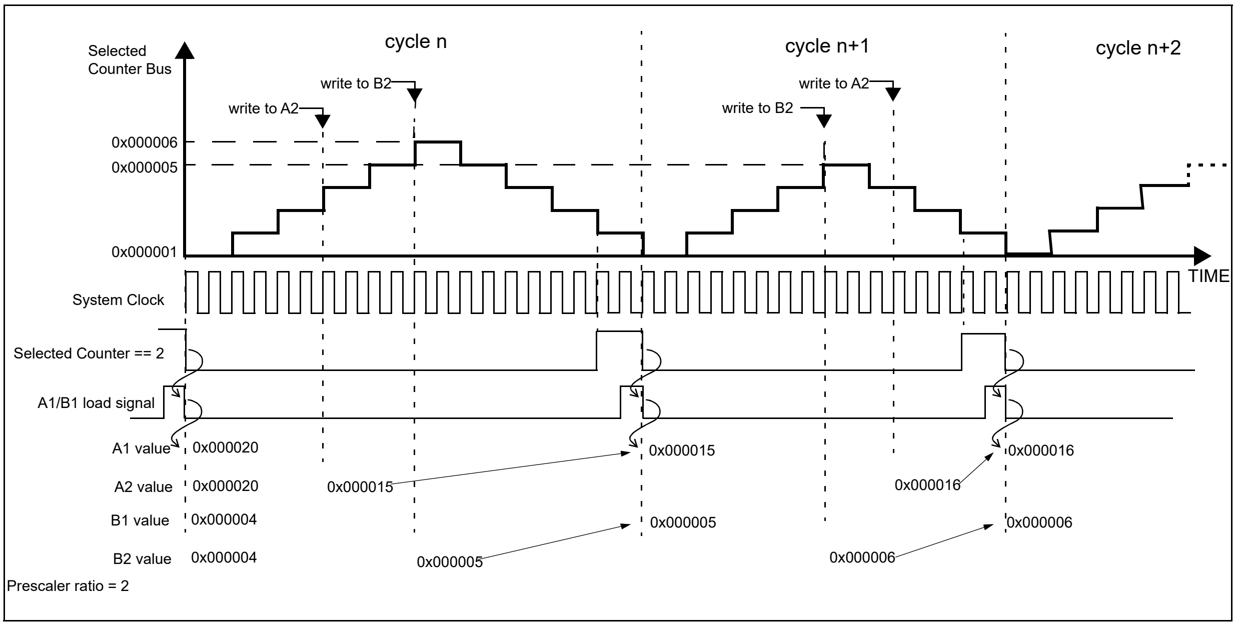
When OPWMCB mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC*n* register.

The following basic steps summarize the proper OPWMCB startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler;
2. *[MCB channel]* Disable Channel Prescaler;
3. *[MCB channel]* Write 0x1 at internal counter;
4. *[MCB channel]* Set A register;
5. *[MCB channel]* Set channel to MCB Up mode;
6. *[MCB channel]* Set prescaler ratio;
7. *[MCB channel]* Enable Channel Prescaler;
8. *[OPWMCB channel]* Disable Channel Prescaler;
9. *[OPWMCB channel]* Set A register;
10. *[OPWMCB channel]* Set B register;
11. *[OPWMCB channel]* Select time base input through BSL[21:22] bits;
12. *[OPWMCB channel]* Enter OPWMCB mode;
13. *[OPWMCB channel]* Set prescaler ratio;
14. *[OPWMCB channel]* Enable Channel Prescaler;
15. *[global]* Enable Global Prescaler.

Figure 502 describes the load of A1 and B1 registers which occurs when the selected counter bus transitions from 0x2 to 0x1. This event defines the cycle boundary. Note that values written to A2 or B2 within cycle **n** are loaded into A1 or B1 registers, respectively, and used to generate matches in cycle **n+1**.

Figure 502. OPWMCB A1 and B1 registers load

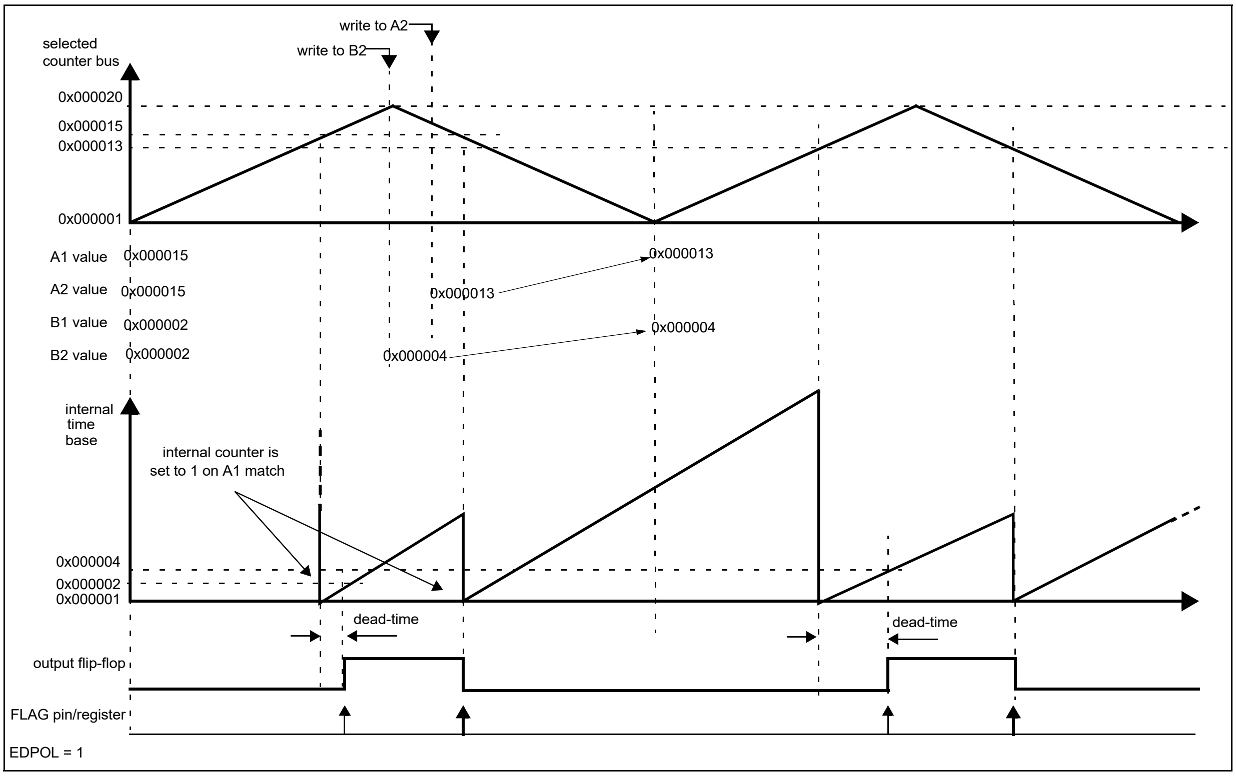


Bit OU*n* of the EMIOSOUDIS register can be used to disable the A1 and B1 updates, thus allowing to synchronize the load on these registers with the load of A1 or B1 registers in others channels. Note that, using the update disable bit A1 and B1 registers can be updated at the same counter cycle thus allowing to change both registers at the same time.

In this mode, A1 matches always set the internal counter to 0x1. When operating with leading edge dead time insertion the first A1 match sets the internal counter to 0x1. When a match occurs between register B1 and the internal time base, the output flip-flop is set to the value of the EDPOL bit. In the following match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously. The internal counter should not reach 0x0 as a consequence of a rollover. In order to avoid it, the user should not write to the EMIOSB register a value greater than twice the difference between external count up limit and EMIOSA value.

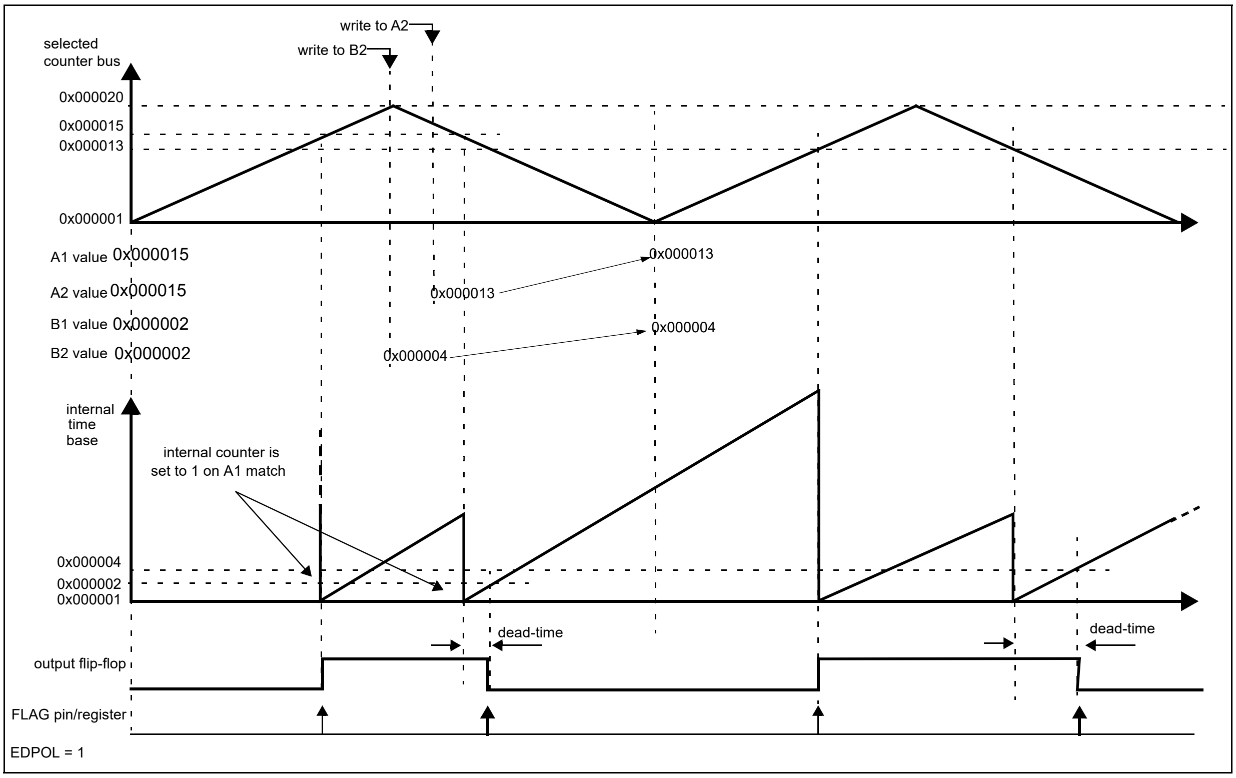
Figure 503 shows two cycles of a Center Aligned PWM signal. Note that, both A1 and B1 register values are changing within the same cycle which allows to vary at the same time the duty cycle and dead time values.

Figure 503. OPWMCB with lead dead time insertion



When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit and sets the internal counter to 0x1. In the second match between register A1 and the selected time base, the internal counter is set to 0x1 and B1 matches are enabled. When the match between register B1 and the selected time base occurs the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

Figure 504. OPWMCB with trail dead time insertion



FLAG can be generated in the trailing edge of the output PWM signal when MODE[30] is cleared, or in both edges, when MODE[30] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

Note: *In OPWMCB mode, FORCMA and FORCMB do not have the same behavior as a regular match. Instead, they force the output flip-flop to constant value which depends upon the selected dead time insertion mode, lead or trail, and the value of the EDPOL bit.*

FORCMA has different behaviors depending upon the selected dead time insertion mode, lead or trail. In lead dead time insertion FORCMA force a transition in the output flip-flop to the opposite of EDPOL. In trail dead time insertion the output flip-flop is forced to the value of EDPOL bit.

If bit FORCMB is set, the output flip-flop value depends upon the selected dead time insertion mode. In lead dead time insertion FORCMB forces the output flip-flop to transition to EDPOL bit value. In trail dead time insertion the output flip-flop is forced to the opposite of EDPOL bit value.

Note: *FORCMA bit set does not set the internal time-base to 0x1 as a regular A1 match.*

The FLAG bit is not set either in case of a FORCMA or FORCMB or even if both forces are issued at the same time.

Note: *FORCMA and FORCMB have the same behavior even in Freeze or normal mode regarding the output pin transition.*

When FORCMA is issued along with FORCMB the output flip-flop is set to the opposite of EDPOL bit value. This is equivalent of saying that.FORCMA has precedence over FORCMB when lead dead time insertion is selected and FORCMB has precedence over FORCMA when trail dead time insertion is selected.

Duty cycle from 0% to 100% can be generated by setting appropriate values to A1 and B1 registers relatively to the period of the external time base. Setting A1 = 1 generates a 100% duty cycle waveform. Assuming EDPOL is set to ‘1’ and OPWMCB mode with trail dead time insertion, 100% duty cycle signals can be generated if B1 occurs at or after the cycle boundary (external counter = 1). If A1 is greater than the maximum value of the selected counter bus period, then a 0% duty cycle is produced, only if the pin starts the current cycle in the opposite of EDPOL value. In case of 100% duty cycle, the transition from EDPOL to the opposite of EDPOL may be obtained by forcing pin, using FORCMA or FORCMB, or both.

Note: If A1 is set to 0x1 at OPWMCB entry the 100% duty cycle may not be obtained in the very first PWM cycle due to the pin condition at mode entry.

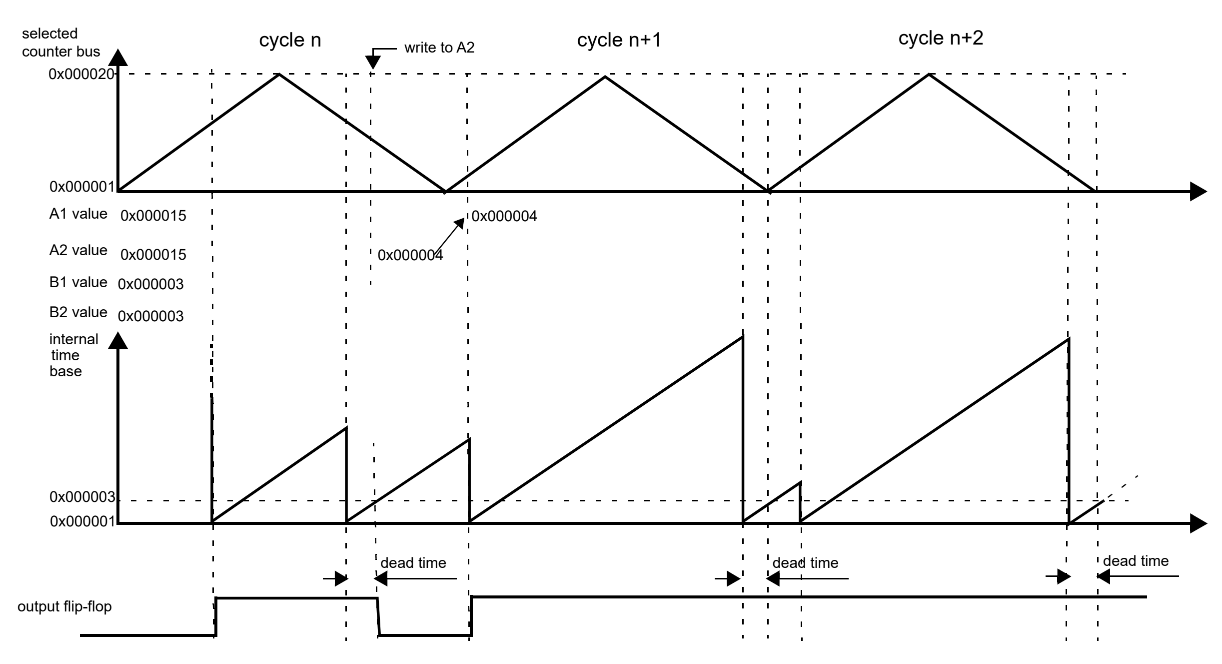
Only values different than 0x0 are allowed to be written to A1 register. If 0x0 is loaded to A1 the results are unpredictable.

Note: A special case occurs when A1 is set to (external counter bus period)/2, which is the maximum value of the external counter. In this case the output flip-flop is constantly set to the EDPOL bit value.

The internal channel logic prevents matches from one cycle to propagate to the next cycle. In trail dead time insertion B1 match from cycle **n** could eventually cross the cycle boundary and occur in cycle **n+1**. In this case B1 match is masked out and does not cause the output flip-flop to transition. Therefore matches in cycle **n+1** are not affected by the late B1 matches from cycle **n**.

Figure 505 shows a 100% duty cycle output signal generated by setting A1 = 4 and B1 = 3. In this case the trailing edge is positioned at the boundary of cycle **n+1**, which is actually considered to belong to cycle **n+2** and therefore does not cause the output flip-flip to transition.

Figure 505. OPWMCB with 100% Duty Cycle (A1 = 4 and B1 = 3)



It is important to notice that, such as in OPWMB and OPWFMB modes, the match signal used to set or clear the channel output flip-flop is generated on the deassertion of the channel combinational comparator output signal which compares the selected time base with A1 or B1 register values. Refer to *Figure 498* which describes the delay from matches to output flip-flop transition in OPWFMB mode. The operation of OPWMCB mode is similar to OPWFMB regarding matches and output pin transition.

45.4.1.1.11 Output Pulse Width Modulation Buffered (OPWMB) Mode

OPWMB mode (MODE[25:31] = 11000b0) is used to generate pulses with programmable leading and trailing edge placement. An external counter driven in MCB Up mode must be selected from one of the counter buses. A1 register value defines the first edge and B1 the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches the selected counter bus and a positive edge occurs when B1 matches the selected counter bus.

The A1 and B1 registers are double buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. Refer to *Figure 500* for more information about A1 and B1 registers update.

FLAG can be generated at B1 matches, when MODE[30] is cleared, or in both A1 and B1 matches, when MODE[30] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A1 or B1 respectively. FLAG bit is not set by the FORCMA and FORCMB operations.

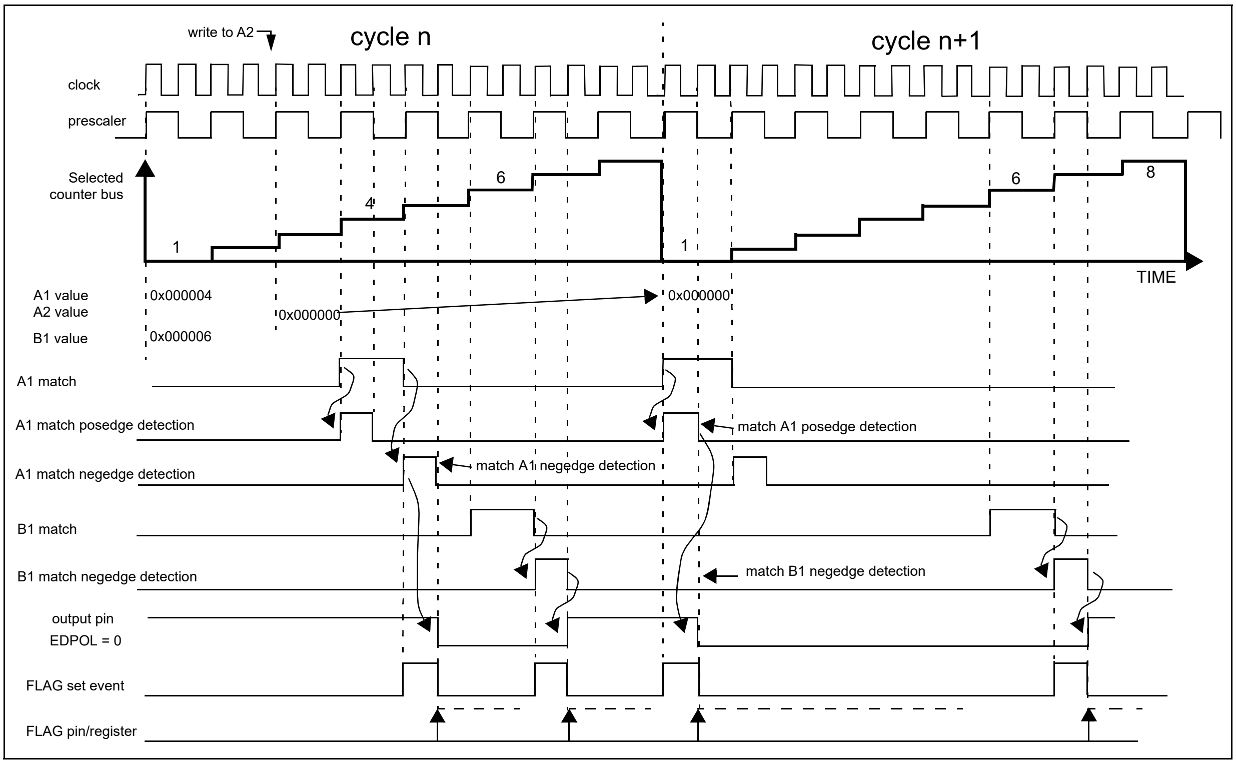
At OPWMB mode entry the output flip-flop is set to the value of the EDPOL bit in the EMIOSC*n* register.

Some rules applicable to the OPWMB mode are:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle
- A1 = 0 match from cycle **n** has precedence over B1 match from cycle **n-1**
- A1 matches are masked out if they occur after B1 match within the same cycle
- Any value written to A2 or B2 on cycle **n** is loaded to A1 and B1 registers at the following cycle boundary (assuming OU*n* bit of EMIOSOUDIS register is not asserted). Thus the new values will be used for A1 and B1 matches in cycle **n+1**

Figure 506 describes the operation of the OPWMB mode regarding A1 and B1 matches and the transition of the channel output pin. In this example EDPOL is set to ‘0’.

Figure 506. OPWMB mode matches and flags



Note that the output pin transitions are based on the negedges of the A1 and B1 match signals. *Figure 506* shows in cycle **n+1** the value of A1 register being set to ‘0’. In this case the match posedge is used instead of the negedge to transition the output flip-flop.

Figure 507 describes the channel operation for 0% duty cycle. Note that the A1 match posedge signal occurs at the same time as the B1 = 0x8 negedge signal. In this case A1 match has precedence over B1 match, causing the output pin to remain at EDPOL bit value, thus generating a 0% duty cycle signal.

Figure 507. OPWMB mode with 0% duty cycle

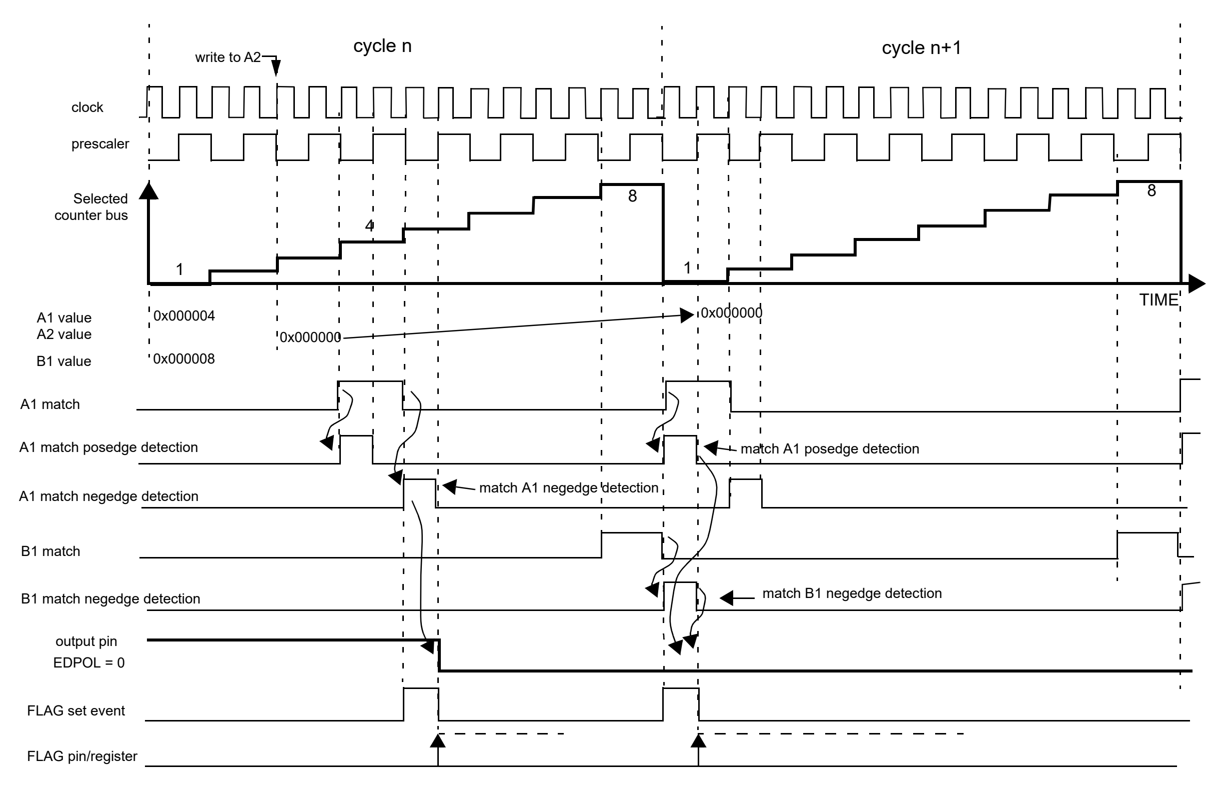
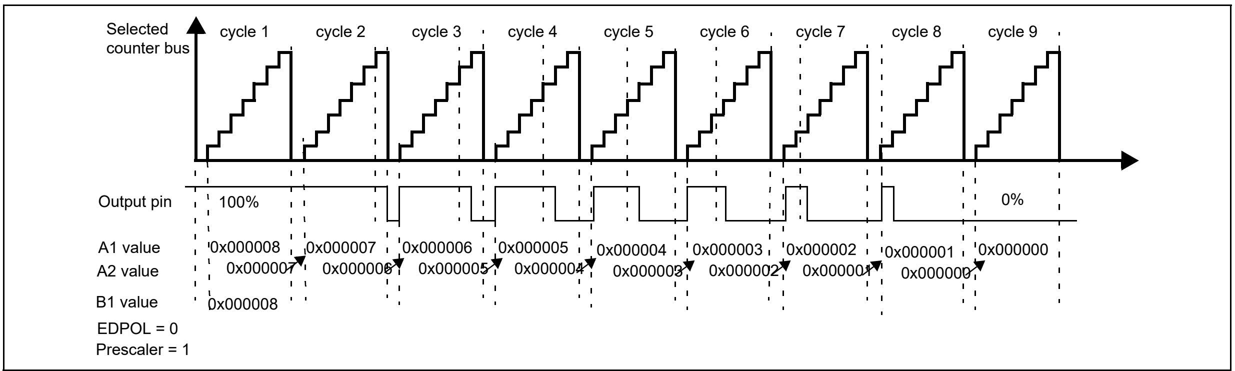


Figure 508 shows a waveform changing from 100% to 0% duty cycle. EDPOL in this case is zero. In this example B1 is programmed to the same value as the period of the external selected time base.

Figure 508. OPWMB mode from 100% to 0% duty cycle



In *Figure 508* if B1 is set to a value lower than 0x8 it is not possible to achieve 0% duty cycle by only changing A1 register value. Since B1 matches have precedence over A1 matches the output pin transitions to the opposite of EDPOL bit at B1 match. Note also that if B1 is set to 0x9, for instance, B1 match does not occur, thus a 0% duty cycle signal is generated.

45.4.1.1.12 Output Pulse Width Modulation with Trigger (OPWMT) mode

OPWMT mode (MODE[25:31] = 0100110) is intended to support the generation of pulse width modulation signals where the period is not modified while the signal is being output, but where the duty cycle will be varied and must not create glitches. The mode is intended to be used in conjunction with other channels executing in the same mode and sharing a common timebase. It will support each channel with a fixed PWM leading edge position with respect to the other channels and the ability to generate a trigger signal at any point in the period that can be output from the module to initiate activity in other parts of the device such as starting ADC conversions.

An external counter driven in either MC Up or MCB Up mode must be selected from one of the counter buses.

Register A1 defines the leading edge of the PWM output pulse and as such the beginning of the PWM's period. This makes it possible to insure that the leading edge of multiple channels in OPWMT mode can occur at a specific time with respect to the other channels when using a shared timebase. This can allow the introduction of a fixed offset for each channel which can be particularly useful in the generation of lighting PWM control signals where it is desirable that edges are not coincident with each other to help eliminate noise generation. The value of register A1 represents the shift of the PWM channel with respect to the selected timebase. A1 can be configured with any value within the range of the selected time base. Note that registers loaded with 0x0 will not produce matches if the timebase is driven by a channel in MCB mode.

A1 is not buffered as the shift of a PWM channel must not be modified while the PWM signal is being generated. In case A1 is modified it is immediately updated and one PWM pulse could be lost.

EMIOSB*n* address gives access to B2 register for write and B1 register for read. Register B1 defines the trailing edge of the PWM output pulse and as such the duty cycle of the PWM signal. To synchronize B1 update with the PWM signal and so ensure a correct output pulse generation the transfer from B2 to B1 is done at every match of register A1.

EMIOSOUDIS register affects transfers between B2 and B1 only.

In order to account for the shift in the leading edge of the waveform defined by register A1 it will be necessary that the trailing edge, held in register B1, can roll over into the next period. This means that a match against the B1 register should not have to be qualified by a match in the A1 register. The impact of this would mean that incorrectly setting register B1 to a value less that register A1 will result in the output being held over a cycle boundary until the B1 value is encountered.

This mode provides a buffered update of the trailing edge by updating register B1 with register B2 contents only at a match of register A1.

The value loaded in register A1 is compared with the value on the selected time base. When a match on comparator A1 occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit.

Note that the output pin and flag transitions are based on the posedges of the A1, B1 and A2 match signals. Refer to *Figure 506* at *Output Pulse Width Modulation Buffered (OPWMB) Mode* for details on match posedge.

Register A2 defines the generation of a trigger event within the PWM period and A2 should be configured with any value within the range of the selected time base, otherwise no trigger will be generated. A match on the comparator will generate the FLAG signal but it has no effect on the PWM output signal generation. The typical setup to obtain a trigger with FLAG is to enable DMA and to drive the channel's ipd_done input high.

A2 is not buffered and therefore its update is immediate. If the channel is running when a change is made this could cause either the loss of one trigger event or the generation of two trigger events within the same period. Register A2 can be accessed by reading or writing the eMIOS UC Alternate A Register (EMIOSALTA) at UC*n* base address +0x14.

FLAG signal is set only at match on the comparator with A2. A match on the comparator with A1 or B1 or B2 has no effect on FLAG.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Any FORCMA and/or FORCMB has priority over any simultaneous match regarding to output pin transitions. Note that the load of B2 content on B1 register at an A match is not inhibited due to a simultaneous FORCMA/FORCMB assertion. If both FORCMA and FORCMB are asserted simultaneously the output pin goes to the opposite of EDPOL value such as if A1 and B1 registers had the same value. FORCMA assertion causes the transfer from register B2 to B1 such as a regular A match, regardless of FORCMB assertion.

If subsequent matches occur on comparators A1 and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

At OPWMT mode entry the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC*n* register.

In order to achieve 0% duty cycle both registers A1 and B must be set to the same value. When a simultaneous match on comparators A and B occur, the output flip-flop is set at every period to the complement value of EDPOL.

In order to achieve 100% duty cycle the register B1 must be set to a value greater than maximum value of the selected time base. As a consequence, if 100% duty cycle must be implemented, the maximum counter value for the time base is 0xFFFE for a 16-bit counter. When a match on comparator A1 occurs the output flip-flop is set at every period to the value of EDPOL bit. The transfer from register B2 to B1 is still triggered by the match at comparator A.

Figure 509 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and duty cycle update on next period update.

Figure 509. OPWMT example

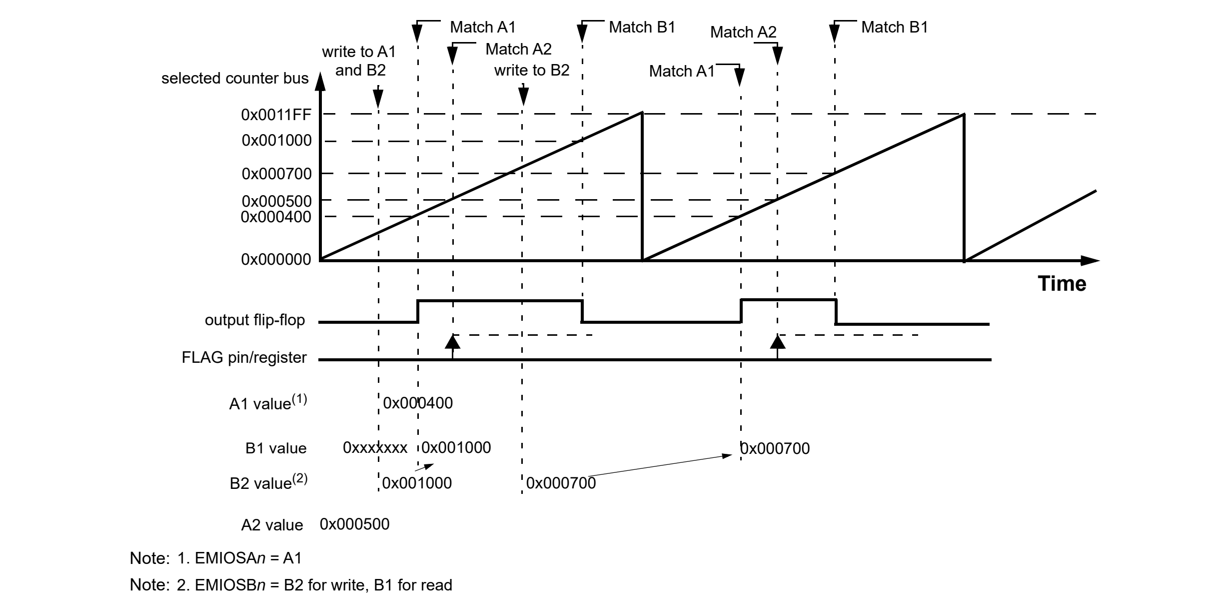


Figure 510 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and 0% duty.

Figure 510. OPWMT with 0% Duty Cycle

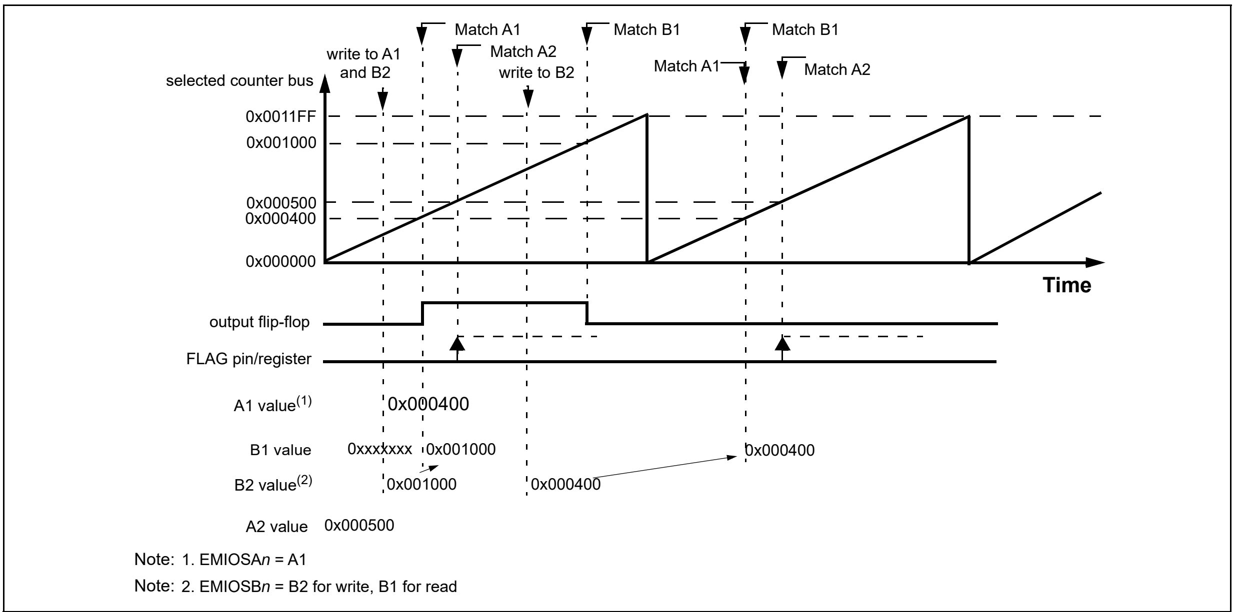
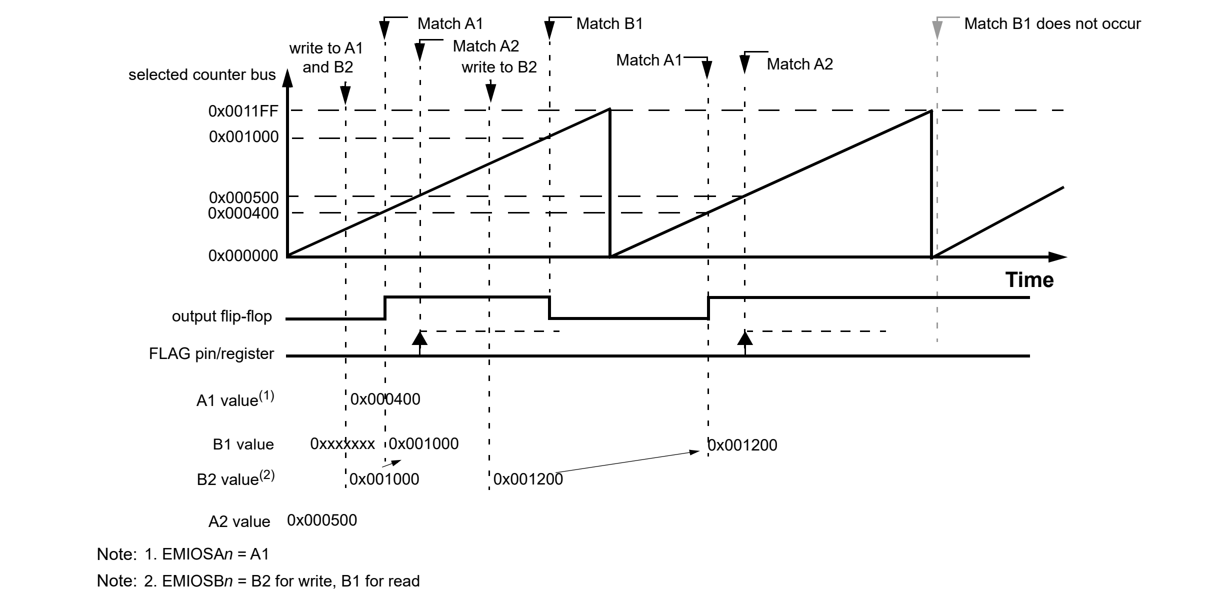


Figure 511 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and 100% duty cycle.

Figure 511. OPWMT with 100% duty cycle

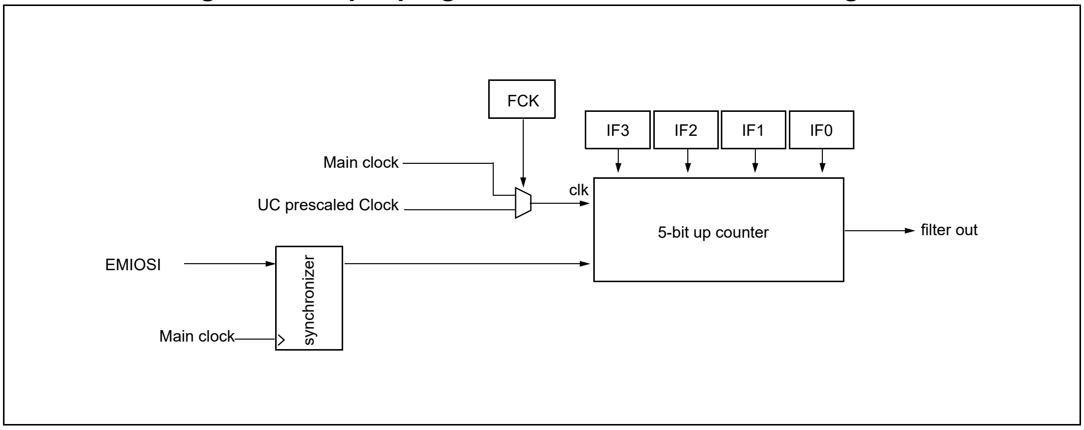


45.4.1.2 Input Programmable Filter (IPF)

The IPF ensures that only valid input pin transitions are received by the Unified Channel edge detector. A block diagram of the IPF is shown in *Figure 512*.

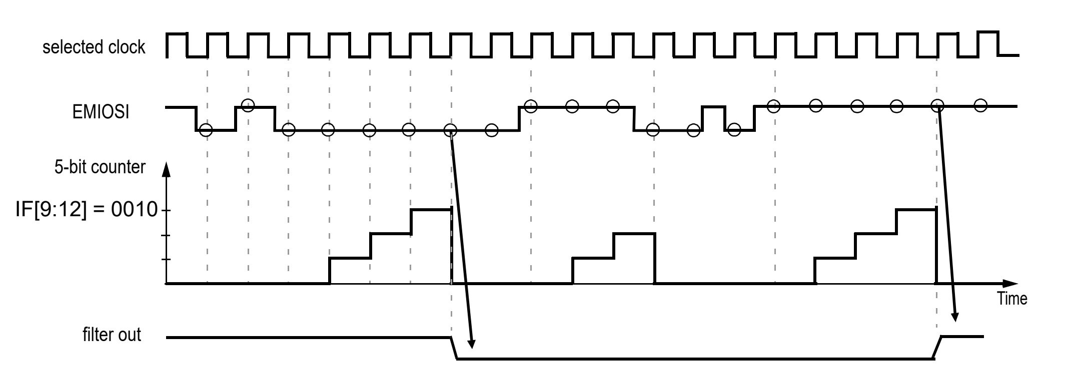
The IPF is a 5-bit programmable up counter that is incremented by the selected clock source, according to bits IF[9:12] in EMIOSC*n* register.

Figure 512. input programmable filter submodule diagram



The input signal is synchronized by the main clock. When a state change occurs in this signal, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If the opposite edge appears on the pin before validation (overflow), the counter is reset. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the edge detector. A timing diagram of the input filter is shown in *Figure 513*.

Figure 513. Input programmable filter example



The filter is not disabled during either freeze state or negated GTBE input.

45.4.1.3 Clock Prescaler (CP)

The CP divides the GCP output signal to generate a clock enable for the internal counter of the Unified Channels. The GCP output signal is prescaled by the value defined in *Figure 580* according to the UCPRE[4:5] bits in EMIOSC*n* register. The prescaler is enabled by setting the UCPREN bit in the EMIOSC*n* and can be stopped at any time by clearing this bit, thereby stopping the internal counter in the Unified Channel.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at both GPREN bit in EMIOSMCR register and UCPREN bit in EMIOSC*n* register, thus disabling prescalers;
2. Write the desired value for prescaling rate at UCPRE[4:5] bits in EMIOSC*n* register;
3. Enable channel prescaler by writing 1 at UCPREN bit in EMIOSC*n* register;
4. Enable global prescaler by writing 1 at GPREN bit in EMIOSMCR register.

The prescaler is not disabled during either freeze state or negated GTBE input.

45.4.1.4 Effect of Freeze on the Unified Channel

When in debug mode, bit FRZ in the EMIOSMCR and bit FREN in the EMIOSC*n* register are both set, the internal counter and Unified Channel capture and compare functions are halted. The UC is frozen in its current state.

During freeze, all registers are accessible. When the Unified Channel is operating in an output mode, the force match functions remain available, allowing the software to force the output to the desired level.

Note that for input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or freeze enable bit is cleared (FRZ in the EMIOSMCR or FREN in the EMIOSC*n* register) the channel actions resume, but may be inconsistent until channel enters GPIO mode again.

45.4.2 IP Bus Interface Unit (BIU)

The BIU provides the interface between the Internal Interface Bus (IIB) and the Peripheral Bus, allowing communication among all submodules and this IP interface.

The BIU allows 8, 16 and 32-bit access. They are performed over a 32-bit data bus in a single cycle clock.

45.4.2.1 Effect of Freeze on the BIU

When the FRZ bit in the EMIOSMCR is set and the module is in debug mode, the operation of BIU is not affected.

45.4.3 Global Clock Prescaler Submodule (GCP)

The GCP divides the system clock to generate a clock for the CPs of the channels. The main clock signal is prescaled by the value defined in *Figure 570* according to bits GPRE[16:23] in the EMIOSMCR. The global prescaler is enabled by setting the GPREN bit in the EMIOSMCR and can be stopped at any time by clearing this bit, thereby stopping the internal counters in all the channels.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write ‘0’ at GPREN bit in EMIOSMCR, thus disabling global prescaler;
2. Write the desired value for prescaling rate at GPRE[16:23] bits in EMIOSMCR;
3. Enable global prescaler by writing ‘1’ at GPREN bit in EMIOSMCR.

The prescaler is not disabled during either freeze state or negated GTBE input.

45.4.3.1 Effect of Freeze on the GCP

When the FRZ bit in the EMIOSMCR is set and the module is in debug mode, the operation of GCP submodule is not affected, that is, there is no freeze function in this submodule.

45.5 Initialization/Application information

On resetting the eMIOS the Unified Channels enter GPIO input mode.

45.5.1 Considerations

Before changing an operating mode, the UC must be programmed to GPIO mode and EMIOSA*n* and EMIOSB*n* registers must be updated with the correct values for the next operating mode. Then the EMIOSC*n* register can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operation cycle of the selected time base can be random, that is, matches can occur in random time if the contents of EMIOSA*n* or EMIOSB*n* were not updated with the correct value before the time base matches the previous contents of EMIOSA*n* or EMIOSB*n*.

When interrupts are enabled, the software must clear the FLAG bits before exiting the interrupt service routine.

45.5.2 Application information

Correlated output signals can be generated by all output operation modes. Bits OU*n* of the EMIOSOUDIS register can be used to control the update of these output signals.

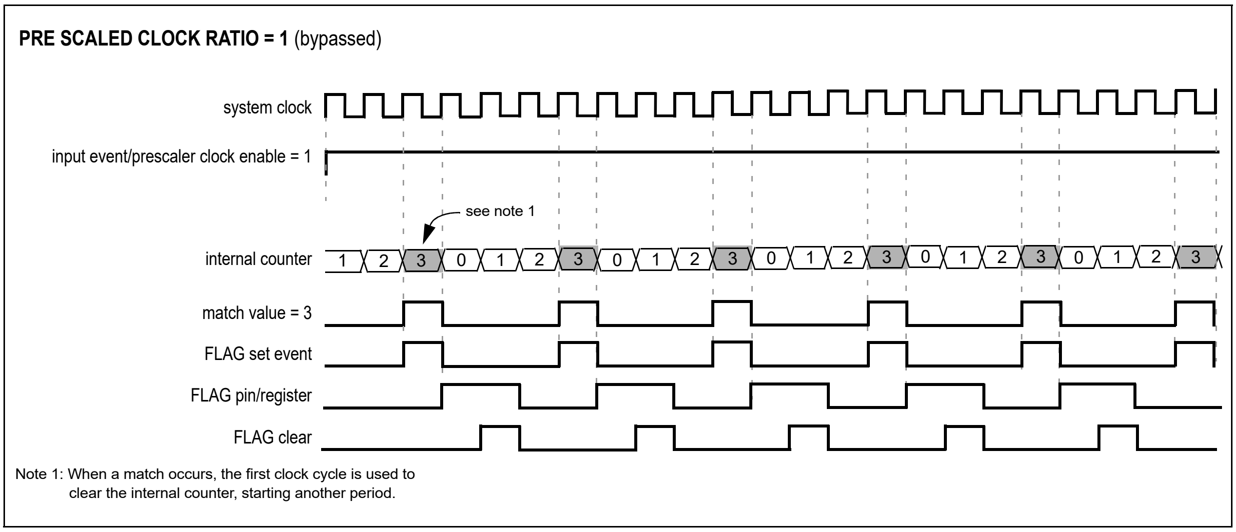
In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global prescaler. If the internal prescalers are set after enabling the global prescaler, the internal counters may increment in the same ratio, but at a different clock cycle.

45.5.2.1 Time base generation

For MC with internal clock source operation modes, the internal counter rate can be modified by configuring the clock prescaler ratio. *Figure 514* shows an example of a time base with prescaler ratio equal to one.

Note: MCB and OPWFMB modes have a different behavior.

Figure 514. Time base period when running in the fastest prescaler ratio

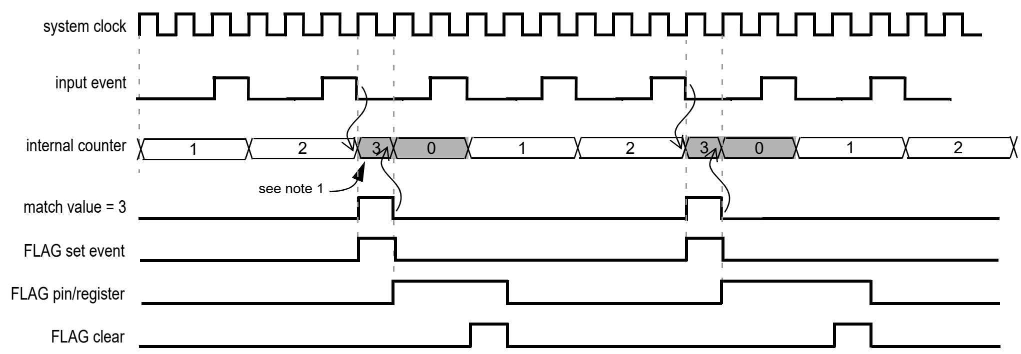


If the prescaler ratio is greater than one or external clock is selected, the counter may behave in three different ways depending on the channel mode:

- If MC mode and Clear on Match Start and External Clock source are selected the internal counter behaves as described in *Figure 515*.
- If MC mode and Clear on Match Start and Internal Clock source are selected the internal counter behaves as described in *Figure 516*.
- If MC mode and Clear on Match End are selected the internal counter behaves as described in *Figure 517*.

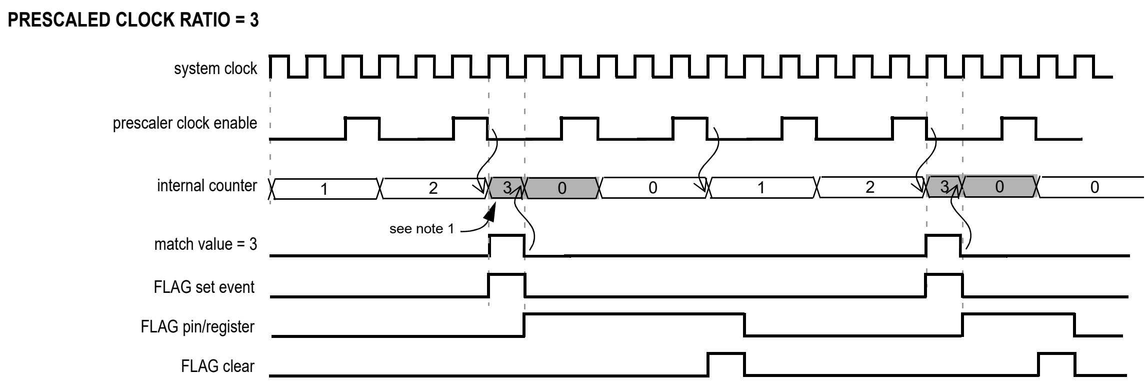
Note: MCB and OPWFMB modes have a different behavior.

Figure 515. Time base generation with external clock and clear on match start



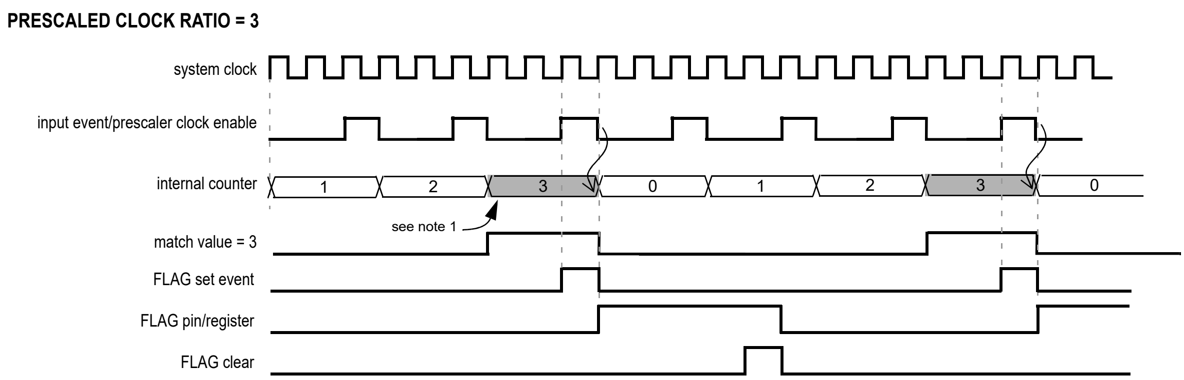
Note 1: When a match occurs, the first system clock cycle is used to clear the internal counter, and at the next edge of prescaler clock enable the counter will start counting.

Figure 516. Time base generation with internal clock and clear on match start



Note 1: When a match occurs, the first clock cycle is used to clear the internal counter, and only after a second edge of pre scaled clock the counter will start counting.

Figure 517. Time base generation with clear on match end



Note 1: The match occurs only when the input event/prescaler clock enable is active. Then, the internal counter is immediately cleared.

45.5.2.2 Coherent accesses

It is highly recommended that the software waits for a new FLAG set event before starting to read the EMIOSA*n* and EMIOSB*n* registers to get a new measurement. The FLAG indicates that new data has been captured and it is the only way to assure data coherency.

The FLAG set event can be detected by polling the FLAG bit or by enabling the interrupt request or CTU trigger generation.

Reading the EMIOSA*n* register again in the same period of the last read of EMIOSB*n* register may lead to incoherent results. This will occur if the last read of EMIOSB*n* register occurred after a disabled B2 to B1 transfer.

45.5.2.3 Channel/Modes initialization

The following basic steps summarize basic output mode startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler.
2. *[timebase channel]* Disable Channel Prescaler.
3. *[timebase channel]* Write initial value at internal counter.
4. *[timebase channel]* Set A/B register.
5. *[timebase channel]* Set channel to MC(B) Up mode.
6. *[timebase channel]* Set prescaler ratio.
7. *[timebase channel]* Enable Channel Prescaler.
8. *[output channel]* Disable Channel Prescaler.
9. *[output channel]* Set A/B register.
10. *[output channel]* Select timebase input through bits BSL[21:22].
11. *[output channel]* Enter output mode.
12. *[output channel]* Set prescaler ratio (same ratio as timebase channel).
13. *[output channel]* Enable Channel Prescaler.
14. *[global]* Enable Global Prescaler.
15. *[global]* Enable Global Time Base.

The timebase channel and the output channel may be the same for some applications such as in OPWFM(B) mode or whenever the output channel is intended to run the timebase itself.

The flags can be configured at any time.

46 CAN subsystem

46.1 Introduction

SPC584Bx has two CAN subsystems implemented:

- CAN Subsystem 0
- CAN Subsystem 1

Note: Refer to Section 7.8.2: CAN subsystem configuration in Chapter 7: Device configuration to see the subsystems implementation.

The Controller Area Network (CAN) subsystem consists of the modular CAN (M_CAN) modules along with an integrated intelligent CAN RAM controller. The CAN RAM controller consists of additional logic for arbitration between the requests for the RAM access by the various CANs and CPU, ECC encoder/decoder for the Message RAM data and active transmit message buffer protection from CPU write access. The subsystem follows the little endian format.

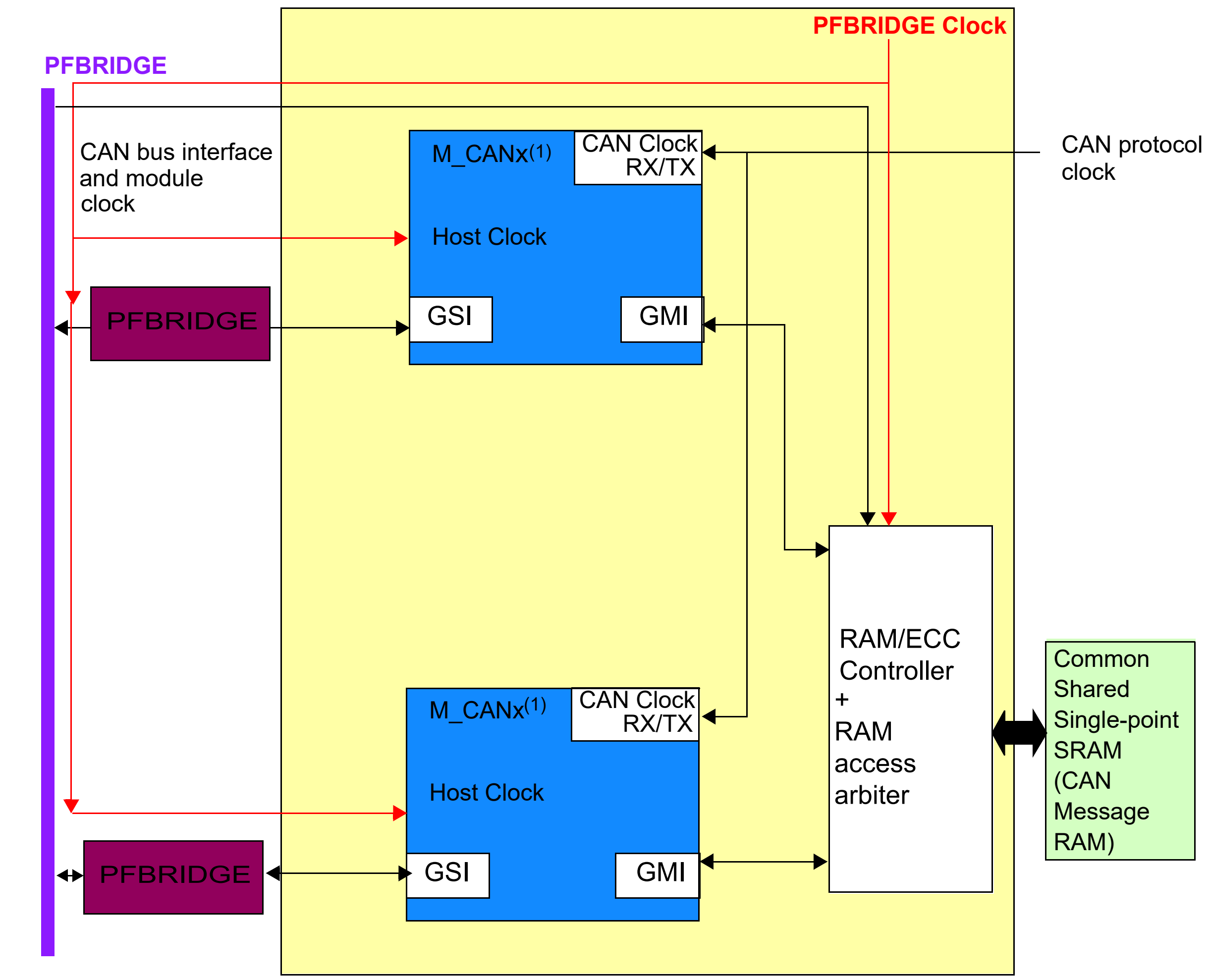
Note: Refer to Section 7.8.2: CAN subsystem configuration in Chapter 7: Device configuration to see the number of CAN nodes used in the device.

Table 586. IP user manual versions

IP Name	Version
M_CAN IP	Revision 3.2.1

The general CAN subsystem implementation is shown in *Section 46.2*. It interfaces with the Host processor bus using the peripheral bus. The RAM is not implemented inside the subsystem.

Figure 518. CAN subsystem generic block diagram



1: Number of M_CAN nodes (M_CANx) varies per device. Please refer to Device Configuration chapter to see the number of M_CAN nodes used in the device.

46.2 Features

The CAN subsystem consists of the following major blocks:

- Modular CAN cores: The registers of the CAN module can be accessed using the Generic Slave Interface (GSI)
- CAN-RAM arbiter
- SRAM interface and memory organization
- ECC Controller

46.3 Modular CAN (M_CAN) cores

M_CAN functionality conforms to CAN specification V2.0B active for each M_CAN node.The M_CAN performs communication according to the CAN protocol specification 2.0 part A,B and to CAN FD 1.0.Flexible assignment of Message Objects to nodes.
The bit rate can be programmed to values up to 1 Mbit/s for standard CAN frames. High bit rates are possible in CAN FD mode. Additional transceiver hardware is required for connection to the physical layer.

Note: Implemented CAN nodes are fully compliant to ISO CAN FD specification ISO 11898-1: 2015.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface. Depending on the chosen device, multiple M_CAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_CAN’s clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

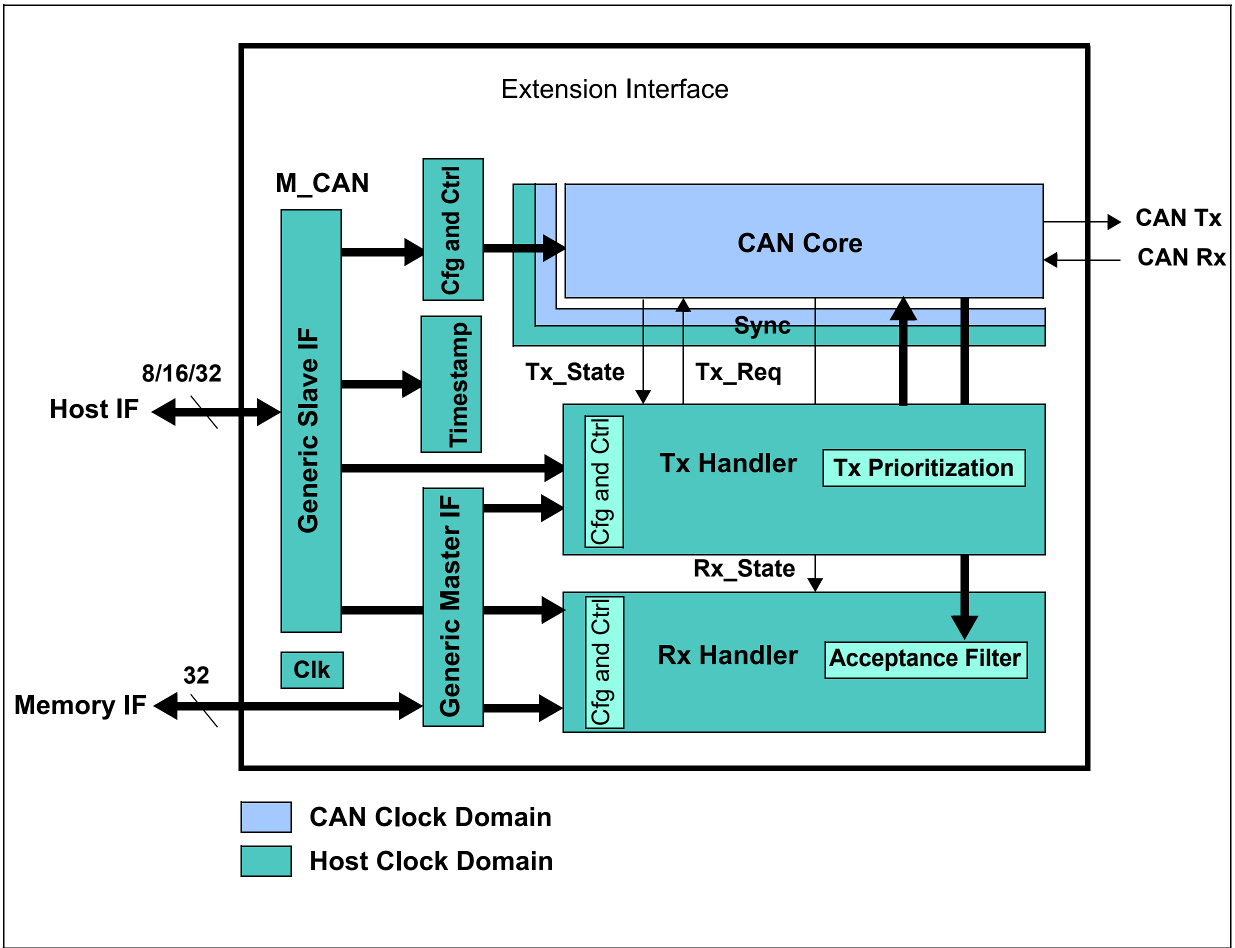
46.3.1 Features

The following are the features of Modular CAN cores.

- Conforms with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015
- CAN Flexible data-rate (ISO CAN FD) protocol with 64 data bytes on M_CAN is supported
- Bit rates up to 1 Mbit/s in standard CAN mode
- Bit rates up to 8 Mbit/s in ISO CAN FD mode
- CAN error logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_CANs share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8-/16-/32-bits Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug over M_CAN
- Clock calibration over M_CAN is supported

46.3.2 Block diagram

Figure 519. M_CAN core block diagram



- CAN core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Clk: Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Cfg and Ctrl: CAN core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.
- Rx Handler: Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored

together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

- Generic Slave Interface: Connects the M_CAN to a customer specific Host CPU. The Generic Slave Interface is capable of connecting to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic Master Interface: Connects the M_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB $\times$ 32 bits.
- Extension Interface: All flags from the Interrupt Register IR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_CAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

46.3.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as CAN clock.

Within the M_CAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note: In order to achieve a stable function of the M_CAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

Caution: Stop this module when the device is in STOP mode. If CAN message reception is mandatory, check that the PLL is unlocked before transmitting any message. Use the external oscillator (XOSC) instead of the internal RC oscillator (IRCOSC) during the STOP mode.

46.3.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to EINT0 or to EINT1. By default all interrupts are routed to interrupt line EINT0. By programming EINT0 and EINT1 bits of the Interrupt Line Enable (ILE) register, the interrupt lines can be enabled or disabled separately.

46.3.5 Memory map and register descriptions

46.3.5.1 Hardware reset description

After hardware reset, the registers of the M_CAN hold the reset values listed in *Table 587*. Additionally the *Bus_Off* state is reset and the M_CAN Tx output is set to *recessive* (HIGH). The value 0x0001 (bit 0 [INIT] of the CCCR register = ‘1’) in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets bit 0 [INIT] of the CCCR register = ‘0’.

46.3.5.2 Register map

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

Note: Write access by the Host CPU to registers/bits marked with "P=Protected Write" is possible only when the bit 1 [CCE] and bit 0 [INIT] of the CCCR register is set to '1'.

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

Caution: Any write access to reserved or not implemented registers in the 16 KB slot assigned by the peripherals bridge to the M_CAN IP will not generate any bus access error.

Table 587. M_CAN memory map

Address Offset	Register Name	Access(1)	Reset value	Location
0x0000	Core Release Register (CREL)	R	0x3213_0506	Section 46.3.5.2.1
0x0004	Endian Register (ENDN)	R	0x8765_4321	Section 46.3.5.2.2
0x0008–0x000B	Reserved			
0x000C	Data Bit Timing and Prescaler Register (DBTP)	RP	0x0000_0A33	Section 46.3.5.2.3
0x0010	Test Register (TEST)	RP	0x0000_0000	Section 46.3.5.2.4
0x0014	RAM Watchdog Register (RWD)	RP	0x0000_0000	Section 46.3.5.2.5
0x0018	CC Control Register (CCCR)	RWPp	0x0000_0001	Section 46.3.5.2.6
0x001C	Nominal Bit Timing and Prescaler Register (NBTP)	RP	0x0600_0A03	Section 46.3.5.2.7
0x0020	Timestamp Counter Configuration Register (TSCC)	RP	0x0000_0000	Section 46.3.5.2.8
0x0024	Timestamp Counter Value Register (TSCV)	RC	0x0000_0000	Section 46.3.5.2.9
0x0028	Timeout Counter Configuration Register (TOCC)	RP	0xFFFF_0000	Section 46.3.5.2.10
0x002C	Timeout Counter Value Register (TOCV)	RC	0x0000_FFFF	Section 46.3.5.2.11
0x0030–0x003F	Reserved			
0x0040	Error Counter Register (ECR)	RX	0x0000_0000	Section 46.3.5.2.12
0x0044	Protocol Status Register (PSR)	RXS	0x0000_0707	Section 46.3.5.2.13
0x0048	Transmitter Delay Compensation Register (TDCR)	RP	0x0000_0000	Section 46.3.5.2.14
0x004C–0x004F	Reserved			
0x0050	Interrupt Register (IR)	R/W	0x0000_0000	Section 46.3.5.2.15
0x0054	Interrupt Enable Register (IE)	R/W	0x0000_0000	Section 46.3.5.2.16
0x0058	Interrupt Line Select Register (ILS)	R/W	0x0000_0000	Section 46.3.5.2.17
0x005C	Interrupt Line Enable Register (ILE)	R/W	0x0000_0000	Section 46.3.5.2.18
0x0060–0x007F	Reserved			
0x0080	Global Filter Configuration Register (GFC)	RP	0x0000_0000	Section 46.3.5.2.19
0x0084	Standard ID Filter Configuration Register (SIDFC)	RP	0x0000_0000	Section 46.3.5.2.20

Table 587. M_CAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x0088	Extended ID Filter Configuration Register (XIDFC)	RP	0x0000_0000	Section 46.3.5.2.21
0x008C–0x008F	Reserved			
0x0090	Extended ID AND Mask Register (XIDAM)	RP	0x1FFF_FFFF	Section 46.3.5.2.22
0x0094	High Priority Message Status Register (HPMS)	R	0x0000_0000	Section 46.3.5.2.23
0x0098	New Data 1 Register (NDAT1)	R/W	0x0000_0000	Section 46.3.5.2.24
0x009C	New Data 2 Register (NDAT2)	R/W	0x0000_0000	Section 46.3.5.2.25
0x00A0	Rx FIFO 0 Configuration Register (RXF0C)	RP	0x0000_0000	Section 46.3.5.2.26
0x00A4	Rx FIFO 0 Status Register (RXF0S)	R	0x0000_0000	Section 46.3.5.2.27
0x00A8	Rx FIFO 0 Acknowledge Register (RXF0A)	R/W	0x0000_0000	Section 46.3.5.2.28
0x00AC	Rx Buffer Configuration Register (RXBC)	RP	0x0000_0000	Section 46.3.5.2.29
0x00B0	Rx FIFO 1 Configuration Register (RXF1C)	RP	0x0000_0000	Section 46.3.5.2.30
0x00B4	Rx FIFO 1 Status Register (RXF1S)	R	0x0000_0000	Section 46.3.5.2.31
0x00B8	Rx FIFO 1 Acknowledge Register (RXF1A)	R/W	0x0000_0000	Section 46.3.5.2.32
0x00BC	Rx Buffer / FIFO Element Size Configuration Register (RXESC)	RP	0x0000_0000	Section 46.3.5.2.33
0x00C0	Tx Buffer Configuration Register (TXBC)	RP	0x0000_0000	Section 46.3.5.2.34
0x00C4	Tx FIFO/Queue Status Register (TXFQS)	R	0x0000_0000	Section 46.3.5.2.35
0x00C8	Tx Buffer Element Size Configuration Register (TXESC)	RP	0x0000_0000	Section 46.3.5.2.36
0x00CC	Tx Buffer Request Pending Register (TXBRP)	R	0x0000_0000	Section 46.3.5.2.37
0x00D0	Tx Buffer Add Request Register (TXBAR)	R/W	0x0000_0000	Section 46.3.5.2.38
0x00D4	Tx Buffer Cancellation Register (TXBCR)	R/W	0x0000_0000	Section 46.3.5.2.39
0x00D8	Tx Buffer Transmission Occurred Register (TXBTO)	R	0x0000_0000	Section 46.3.5.2.40
0x00DC	Tx Buffer Cancellation Finished Register (TXBCF)	R	0x0000_0000	Section 46.3.5.2.41
0x00E0	Tx Buffer Transmission Interrupt Enable (TXBTIE)	R/W	0x0000_0000	Section 46.3.5.2.42
0x00E4	Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)	R/W	0x0000_0000	Section 46.3.5.2.43
0x00E8–0x00EF	Reserved			
0x00F0	Tx Event FIFO Configuration Register (TXEFC)	RP	0x0000_0000	Section 46.3.5.2.44
0x00F4	Tx Event FIFO Status Register (TXEFS)	R	0x0000_0000	Section 46.3.5.2.45

Table 587. M_CAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x00F8	Tx Event FIFO Acknowledge Register (TXEFA)	R/W	0x0000_0000	Section 46.3.5.2.46
0x00FC–0x00FF	Reserved			

1. R = Read, S = Set on read, W = Write, P = Protected Write, C = Clear/preset on write, p = Protected set, X= Reset on read

46.3.5.2.1 Core Release Register (CREL)

Offset: 0x0000 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset(1)	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-d	-d	-d	-d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset(1)	-d	d-	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d

1. -r = release, -d = time stamp. The coding of revisions depends on the module version used in the device. For the device reset value refer to *Table 587: M_CAN memory map*.

Figure 520. Core Release Register (CREL)

Table 588. CREL field descriptions

Field	Description
0:3 REL	Core Release One digit, BCD.
4:7 STEP	Step of Core Release One digit, BCD.
8:11 SUBSTEP	Substep of Core Release One digit, BCD.
12:15 YEAR	Time Stamp Year One digit, BCD.
16:23 MON	Time Stamp Month Two digits, BCD.
24:31 DAY	Time Stamp Day Two digits, BCD.

46.3.5.2.2 Endian Register (ENDN)

Offset: 0x0004 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETV															
W																
Reset	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETV															
W																
Reset	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1

Figure 521. Endian Register (ENDN)

Table 589. ENDN field descriptions

Field	Description
0:31 ETV	Endianness Test Value The endianness test value is 0x87654321.

46.3.5.2.3 Data Bit Timing and Prescaler Register (DBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programmed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 M_CAN clock periods. tq = (FBRP + 1) M_CAN clock period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. **FTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) **[FTSEG1 + FTSEG2 + 3]** tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning that the data for the next bit is available at the first clock edge after the sample point.

Offset: 0x000C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TDC(1)	0	0	DBRP(1)				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	DTSEG1(1)					DTSEG2(1)				DSJW(1)			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 522. Data Bit Timing and Prescaler Register (DBTP)

Table 590. DBTP field descriptions

Field	Description
8 TDC	Transceiver Delay Compensation 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
11:15 DBRP	Data Baud Rate Prescaler (0x00–0x1F) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
19:23 DTSEG1	Data time segment before sample point (0x0–0x1F) Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24:27 DTSEG2	Data time segment after sample point (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28:31 DSJW	Data (Re) Synchronization Jump Width (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 kbit/s.

The bit rate configured for the CAN FD data phase via DBTP must be higher than or equal to the bit rate configured for the arbitration phase via NBTP.

To achieve high bit rates in CAN FD mode for the data phase up to 8 Mbit/s the number of time quanta bits per CAN bit can be chosen down to 4 time quanta per bit time.

46.3.5.2.4 Test Register (TEST)

Write access to the Test Register has to be enabled by setting CCCR[TEST] to ‘1’. All Test Register functions are set to their reset values when CCCR[TEST] is reset. Loop Back mode and software control of M_CAN Tx pin are hardware test modes. Programming of **TX** ≠ ‘00’ may disturb the message transfer on the CAN bus.

Offset: 0x0010 Access: Read-Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RX	TX(1)		LBCK(1)	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 523. Test Register (TEST)

Table 591. TEST field descriptions

Field	Description
24 RX	Receive Pin Monitors the actual value of CAN Rx pin. 0 The CAN bus is dominant (CAN Rx input = '0') 1 The CAN bus is recessive (CAN Rx input = '1')
25:26 TX	Control of Transmit Pin 00 Reset value, CAN Tx output is controlled by the CAN core, updated at the end of the CAN bit time. 01 Sample Point can be monitored at pin CAN Tx output 10 Dominant ('0') level at pin CAN Tx output 11 Recessive ('1') at pin CAN Tx output
27 LBCK	Loop Back mode 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (refer to *Section 46.3.7.9: Test modes*)

46.3.5.2.5 RAM Watchdog Register (RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the WDC bits. The counter is reloaded with WDC bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag WDI bit of the IR is set. The RAM Watchdog Counter is clocked by the Host clock .

Offset: 0x0014 Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDV								WDC(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 524. RAM Watchdog Register (RWD)

Table 592. RWD field descriptions

Field	Description
16:23 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value.
24:31 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of ‘00’ the counter is disabled.

46.3.5.2.6 CC Control Register (CCCR)

The following section shows the CAN Control register.For details about setting and resetting of single bits refer to *Section 46.3.7.1.1: Software initialization*.

Offset: 0x0018 Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NISO(1)	TXP(1)	EFBI(1)	PXHD(1)	0	0	BRSE(1)	FDOE(1)	TEST(2)	DAR(1)	MON(2)	CSR	CSA	ASM(2)	CCE(1)	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.
2. These are protected set (p) bits.

Figure 525. CC Control register (CCCR)

Table 593. CCCR field descriptions

Field	Description
16 NISO	Non ISO Operation If this bit is set, the M_CAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. 0 CAN FD frame format according to ISO11898-1 1 CAN FD frame format according to Bosch CAN FD Specification V1.0
17 TXP	Transmit Pause If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame. 0 Transmit pause disabled 1 Transmit pause enabled
18 EFBI	Edge Filtering during Bus Integration 0 Edge filtering disabled 1 Two consecutive dominant tq required to detect an edge for hard synchronization
19 PXHD	Protocol Exception Handling Disable 0 Protocol exception handling enabled 1 Protocol exception handling disabled **Note:** When protocol exception handling is disabled, the M_CAN will transmit an error frame when it detects a protocol exception condition.
22 BRSE	Bit Rate Switch Enable 0 Bit rate switching for transmissions disabled 1 Bit rate switching for transmissions enabled **Note:** When CAN FD operation is disabled FDOE = ‘0’, BRSE is not evaluated.
23 FDOE	FD Operation Enable 0 FD operation disabled 1 FD operation enabled
24 TEST	Test Mode Enable 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	Disable Automatic Retransmission 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode Bit MON can only be set by the Host when both CCE and INIT are set to ‘1’. The bit can be reset by the Host at any time. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_CAN may be set in power down by stopping Host clock and core clock

Table 593. CCCR field descriptions (continued)

Field	Description
29 ASM	Restricted Operation Mode Bit ASM is only set by the Host when both CCE and INIT are set to ‘1’. The bit is reset by the Host at any time. 0 Normal CAN operation 1 Restricted Operation Mode active
30 CCE	Configuration Change Enable 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = ‘1’)
31 INIT	Initialization 0 Normal Operation 1 Initialization is started **Note:** Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

46.3.5.2.7 Nominal Bit Timing and Prescaler Register (NBTP)

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programmed in the range of [4…385] time quanta. The CAN time quantum may be programmed in the range of [1…512] M_CAN clock periods.

Equation 30 **tq = (NBRP + 1) M_CAN clock period**

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Offset: 0x001C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	NSJW(1)								NBRP(1)							
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	NTSEG1(1)								0	NTSEG2(1)						
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 526. Nominal Bit Timing and Prescaler Register (NBTP)

Table 594. NBTP field descriptions

Field	Description
0:6 NSJW	(Re) Synchronization Jump Width (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
7:15 NBRP	Baud Rate Prescaler (0x000–0x1FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16:23 NTSEG1	Time segment before sample point (0x01–0xFF) Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
25:31 NTSEG2	The time segment after the sample point (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x06000A03 configures the M_CAN for a bit rate of 500 kbit/s.

46.3.5.2.8 Timestamp Counter Configuration Register (TSCC)

Offset: 0x0020 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TCP(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TSS(1)	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 527. Timestamp Counter Configuration Register (TSCC)

Table 595. TSCC field descriptions

Field	Description
12:15 TCP	Timestamp Counter Prescaler (0x0–0xF) Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
30:31 TSS	Timestamp Select 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to Timestamp Counter Prescaler (TCP) 10 Reserved 11 Same as ‘00’ **Note:** In case of CAN FD mode the absolute duration of a bit time does change in one CAN message due to the different baud rates for arbitration phase and data phase. Therefore the timestamp counter does still reflect increasing number on bit times but the fix relationship to absolute time is not given any more. Anyway, it is possible to establish in which order the frames are received.

46.3.5.2.9 Timestamp Counter Value Register (TSCV)

Offset: 0x0024 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 528. Timestamp Counter Value Register (TSCV)

Table 596. TSCV field descriptions

Field	Description
16:31 TSC	Timestamp Counter The internal Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC.TSS = ‘01’, the Timestamp Counter is incremented in multiples of CAN bit times [1 to 16] depending on the configuration of TSCC.TCP. A wrap around sets interrupt flag IR.TSW. Write access resets the counter to zero.

Note: A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

46.3.5.2.10 Timeout Counter Configuration Register (TOCC)

For a description of the Timeout Counter, refer to *Section 46.3.9*.

Offset: 0x0028 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TOP(1)															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TOS(1)		ETOC(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 529. Timeout Counter Configuration Register (TOCC)

Table 597. TOCC field descriptions

Field	Description
0:15 TOP	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
29:30 TOS	Timeout Select When operating in Continuous mode, a write to **TOCV** presets the counter to the value configured by **TOCC[TOP]** and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCC[TOP]**. Down-counting is started when the first FIFO element is stored. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counter 0 Timeout Counter disabled 1 Timeout Counter enabled **Note:** For use of timeout function with CAN FD refer to *Section 46.3.9*.

46.3.5.2.11 Timeout Counter Value Register (TOCV)

Offset: 0x002C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TOC															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 530. Timeout Counter Value Register (TOCV)

Table 598. TOCV field descriptions

Field	Description
16:31 TOC	The Timeout Counter is decremented in multiples of CAN bit times (1 to 16) depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

46.3.5.2.12 Error Counter Register (ECR)

Offset: 0x0040 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CEL(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Access type is RX: reset on read.

Figure 531. Error Counter Register (ECR)

Table 599. ECR field descriptions

Field	Description
8:15 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag ELO bit of the IR register.
16 RP	Receive Error Passive 0 The Receive Error Counter is below the error passive level of 128 1 The Receive Error Counter has reached the error passive level of 128
17:23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127.
24:31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255.

Note: When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

46.3.5.2.13 Protocol Status Register (PSR)

Offset: 0x0044 Access: RXS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	TDCV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PXE[1]	RFDF[1]	RBRS[1]	RESI[1]	DLEC[2]			BO	EW	EP	ACT		LEC[2]		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

1. Access type is RX: reset on read.
2. Access type is RS: set on read.

Figure 532. Protocol Status Register (PSR)

Table 600. PSR field descriptions

Field	Description
9:15 TDCV	Transmitter Delay Compensation Value (0x00-0x7F) Position of the secondary sample point, defined by the sum of the measured delay from CAN Tx to CAN Rx and TDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.
17 PXE	Protocol Exception Event 0 No protocol exception event occurred since last read access 1 Protocol exception event occurred
18 RFDF	Received CAN FD Message This bit is set independent of acceptance filtering. 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with FDF flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with RFDF, independent of acceptance filtering. 0 Last received CAN FD message did not have its BRS flag set. 1 Last received CAN FD message had its BRS flag set.
20 RESI	ESI CAN FD Message with ESI flag This bit is set together with RFDF, independent of acceptance filtering. 0 Last received CAN FD message did not have its ESI flag set. 1 Last received CAN FD message had its ESI flag set
21:23 DLEC	Data Phase Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.
24 BO	Bus_Off Status 0 The M_CAN is not Bus_Off 1 The M_CAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_CAN is in the Error_Passive state

Table 600. PSR field descriptions (continued)

Field	Description
27:28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing – node is synchronizing on CAN communication 01 Idle-node is neither receiver nor transmitter 10 Receiver-node is operating as receiver 11 Transmitter-node is operating as transmitter
29:31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_CAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

The Bus_Off recovery sequence (refer to CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT.] If the device goes Bus_Off, it will set CCCR[INIT] of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 $\times$ 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit 0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

46.3.5.2.14 Transmitter Delay compensation Register (TDCR)

Offset: 0x0048 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TDCO(1)							0	TDCF(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 533. Transmitter Delay compensation Register (TDCR)

Table 601. TDCR field descriptions

Field	Description
17:23 TDCO	Transmitter Delay Compensation Offset (0x00-0x7F) Offset value defining the distance between the measured delay from CAN Tx to CAN Rx and the secondary sample point. Valid values are 0 to 127 mtq.
25:31 TDCF	Transmitter Delay Compensation Filter Window Length (0x00-0x7F) Defines the minimum value for the SSP position, dominant edges on CAN Rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq.

46.3.5.2.15 Interrupt Register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Offset: 0x0050 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 534. Interrupt Register (IR)

Table 602. IR field descriptions

Field	Description
2 ARA	Access to Reserved Address 0 No access to reserved address occurred 1 Access to reserved address occurred
3 PED	Protocol Error in Data Phase (Data Bit Time is used) 0 No protocol error in data phase 1 Protocol error in data phase detected (PSR.DLEC ≠ 0,7)
4 PEA	Protocol Error in Arbitration Phase (Nominal Bit Time is used) 0= No protocol error in arbitration phase 1= Protocol error in arbitration phase detected (PSR.LEC ≠ 0,7)
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred

Table 602. IR field descriptions (continued)

Field	Description
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to ‘1’. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (for example parity logic)
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (for example ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 MRAF	Message RAM Access Failure The flag is set, when the Rx Handler – has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. – was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler is not able to read a message from the Message RAM in time. In this case message transmission is aborted. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM. 0 No Message RAM access failure occurred 1 Message RAM access failure occurred
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Event Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full

Table 602. IR field descriptions (continued)

Field	Description
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark
19 TEFN	Tx Event FIDO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

46.3.5.2.16 Interrupt Enable Register (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt register will be signaled on an interrupt line.

Offset: 0x0054 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 535. Interrupt Enable Register (IE)

Table 603. IE field descriptions

Field	Description
2 ARAE	Access to Reserved Address Enable 0 Interrupt disabled 1 Interrupt enabled
3 PEDE	Protocol Error in Data Phase Enable 0 Interrupt disabled 1 Interrupt enabled
4 PEAE	Protocol Error in Arbitration Phase Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 603. IE field descriptions (continued)

Field	Description
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 MRAFE	Message RAM Access Failure Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 603. IE field descriptions (continued)

Field	Description
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

46.3.5.2.17 Interrupt Line Select Register (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

Offset: 0x0058 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 536. Interrupt Line Select Register (ILS)

Table 604. ILS field descriptions

Field	Description
2 ARAL	Access to Reserved Address Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
3 PEDL	Protocol Error in Data Phase Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
4 PEAL	Protocol Error in Arbitration Phase Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
14 MRAFL	Message RAM Access Failure Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 604. ILS field descriptions (continued)

Field	Description
16 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 604. ILS field descriptions (continued)

Field	Description
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

46.3.5.2.18 Interrupt Line Enable Register (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1

Offset: 0x005C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EINT1	EINT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 537. Interrupt Line Enable Register (ILE)

Table 605. ILE field descriptions

Field	Description
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

46.3.5.2.19 Global Filter Configuration Register (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in *Figure 576* and *Figure 577*.

Offset: 0x0080 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ANFS(1)		ANFE(1)		RRFS(1)	RRFE(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 538. Global Filter Configuration Register (GFC)

Table 606. GFC field descriptions

Field	Description
26:27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28:29 ANFE	ANFE[1:0]: Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

46.3.5.2.20 Standard ID Filter Configuration Register (SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in *Figure 570*.

Offset: 0x0084 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LSS[1]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLSSA[1]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 539. Standard ID Filter Configuration register (SIDFC)

Table 607. SIDFC field descriptions

Field	Description
8:15 LSS	List Size Standard 0x00: No standard Message ID filter 0x01: Number of standard Message ID filter elements ... 0x80: Number of standard Message ID filter elements 0x81: Values greater than 128 are interpreted as 128 ... 0xFF: Values greater than 128 are interpreted as 128
16:29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, refer to *Figure 566*).

46.3.5.2.21 Extended ID Filter Configuration Register (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in *Figure 571*.

Offset: 0x0088 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LSE[1]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLESA[1]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 540. Extended ID Filter Configuration register (XIDFC)

Table 608. XIDFC field descriptions

Field	Description
9:15 LSE	List Size Extended 0x00: No extended Message ID filter 0x01: Number of extended Message ID filter elements ... 0x40: Number of extended Message ID filter elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64
16:29 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, refer to *Figure 566*).

46.3.5.2.22 Extended ID and Mask Register (XIDAM)

Offset: 0x0090 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	EIDM(1)												
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIDM(1)															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 541. Extended ID and Mask Register (XIDAM)

Table 609. XIDAM field descriptions

Field	Description
3:31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

46.3.5.2.23 High Priority Message Status Register (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Offset: 0x0094 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 542. High Priority Message Status Register (HPMS)

Table 610. HPMS field descriptions

Field	Description
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17:23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24:25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26:31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = ‘1’.

46.3.5.2.24 New Data 1 Register (NDAT1)

Offset: 0x0098 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 543. New Data 1 Register (NDAT1)

Table 611. NDAT1 field descriptions

Field	Description
0:31 ND[31:0]	New Data[31:0] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

46.3.5.2.25 New Data 2 Register (NDAT2)

Offset: 0x009C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 544. New Data 2 Register (NDAT2)

Table 612. NDAT2 field descriptions

Field	Description
0:31 ND[63:32]	New Data[63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

46.3.5.2.26 Rx FIFO 0 Configuration Register (RXF0C)

Offset: 0x00A0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F0OM	F0WM[(1)]							0	F0S[(1)]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F0SA[(1)]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 545. Rx FIFO 0 Configuration Register (RXF0C)

Table 613. RXF0C field descriptions

Field	Description
0 F0OM	FIFO 0 Operation Mode FIFO 0 can be operated in blocking or in overwrite mode. 0 FIFO 0 blocking mode 1 FIFO 0 overwrite mode
1:7 F0WM	Rx FIFO 0 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) 0x40: Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) 0x41: Watermark interrupt disabled 0x7F: Watermark interrupt disabled
9:15 F0S	Rx FIFO 0 Size 0x00: No Rx FIFO 0 0x01: Number of Rx FIFO 0 elements ... 0x40: Number of Rx FIFO 0 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1.
16:29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, *Figure 566*).

46.3.5.2.27 Rx FIFO 0 Status Register (RXF0S)

Offset: 0x00A4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RF0L	F0F	0	0	F0PI					
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F0GI						0	F0FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 546. Rx FIFO 0 Status Register (RXF0S)

Table 614. RXF0S field descriptions

Field	Description
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero **Note:** Overwriting the oldest message when RXF0C.F0OM = ‘1’ will not set this flag.
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
10:15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.
18:23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
25:31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

46.3.5.2.28 Rx FIFO 0 Acknowledge Register (RXF0A)

Offset: 0x00A8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F0AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 547. Rx FIFO 0 Acknowledge Register (RXF0A)

Table 615. RXF0A field descriptions

Field	Description
26:31 F0AI	Rx FIFO 0 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL].

46.3.5.2.29 Rx Buffer Configuration Register (RXBC)

Offset: 0x00AC Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 548. Rx Buffer Configuration Register (RXBC)

Table 616. RXBC field descriptions

Field	Description
16:29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.

46.3.5.2.30 Rx FIFO 1 Configuration Register (RXF1C)

Offset: 0x00B0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F1OM(1)	F1WM(1)							0	F1S(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F1SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 549. Rx FIFO 1 Configuration Register (RXF1C)

Table 617. RXF1C field descriptions

Field	Description
0 F1OM	FIFO 1 Operation Mode FIFO 1 can be operated in blocking or in overwrite mode. 0 FIFO 1 blocking mode 1 FIFO 1 overwrite mode
1:7 F1WM	Rx FIFO 1 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 1 watermark interrupt IR[RF1W] ... 0x40: Level for Rx FIFO 1 watermark interrupt IR[RF1W] 0x41: Watermark interrupt disabled ... 0x7F: Watermark interrupt disabled
9:15 F1S	F1S[6:0]: Rx FIFO 1 Size 0x00: No Rx FIFO 1 0x01: Number of Rx FIFO 1 elements ... 0x40: Number of Rx FIFO 1 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1
16:29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, refer to *Figure 566*).

46.3.5.2.31 Rx FIFO 1 Status Register (RXF1S)

Offset: 0x00B4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS		0	0	0	0	RF1L	F1F	0	0	F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 550. Rx FIFO 1 Status Register (RXF1S)

Table 618. RXF1S field descriptions

Field	Description
0:1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
6 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero **Note:** Overwriting the oldest message when RXF1C.F1OM = ‘1’ will not set this flag.
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
10:15 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
18:23 F1GI	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
25:31 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

46.3.5.2.32 Rx FIFO 1 Acknowledge Register (RXF1A)

Offset: 0x00B8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F1AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 551. Rx FIFO 1 Acknowledge Register (RXF1A)

Table 619. RXF1A field descriptions

Field	Description
26:31 F1AI	Rx FIFO 1 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

46.3.5.2.33 Rx Buffer / FIFO Element Size Configuration (RXESC)

It configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

Offset: 0x00BC Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RBDS(1)				F1DS(1)				F0DS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 552. Rx Buffer / FIFO Element Size Configuration Register (RXESC)

Table 620. RXESC field descriptions

Field	Description
21:23 RBDS	Rx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
25:27 F1DS	Rx FIFO 1 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
29:31 F0DS	Rx FIFO 0 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

Note: In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by RXESC is stored to the Rx Buffer respectively, Rx FIFO element. The rest of the frame's data field is ignored.

46.3.5.2.34 Tx Buffer Configuration Register (TXBC)

Offset: 0x00C0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TFQM(1)	TFQS(1)						0	0	NDTB(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 553. Tx Buffer Configuration Register (TXBC)

Table 621. TXBC field descriptions

Field	Description
1 TFQM	Tx FIFO/Queue Mode 0 Tx FIFO operation 1 Tx Queue operation
2:7 TFQS	Tx FIFO/Queue Size 0x00: No Tx FIFO/Queue 0x01: Number of Tx Buffers used for Tx FIFO/Queue ... 0x20: Number of Tx Buffers used for Tx FIFO/Queue 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
10:15 NDTB	Number of Dedicated Transmit Buffers 0x00: No Dedicated Tx Buffers 0x01: Number of Dedicated Tx Buffers ... 0x20: Number of Dedicated Tx Buffers 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
16:29 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, refer to *Figure 566*).

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

46.3.5.2.35 Tx FIFO/Queue Status Register (TXFQS)

Offset: 0x00C4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TFGI					0	0	TFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 554. Tx FIFO/Queue Status Register (TXFQS)

Table 622. TXFQS field descriptions

Field	Description
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11:15 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31.
19:23 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).
26:31 TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get indexes indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

46.3.5.2.36 Tx Buffer Element Size Configuration (TXESC)

It configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

Offset: 0x00C8 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TBDS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 555. Tx Buffer Element Size Configuration Register (TXESC)

Table 623. TXESC field descriptions

Field	Description
29:31 TBDS	Tx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field **Note:** In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as ‘0xCC’ (padding bytes).

46.3.5.2.37 Tx Buffer Request Pending Register (TXBRP)

Offset: 0x00CC Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 556. Tx Buffer Request Pending Register (TXBRP)

Table 624. TXBRP field descriptions

Field	Description
0:31 TRP[31:0]	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF – after successful transmission together with the corresponding TXBTO bit – when the transmission has not yet been started at the point of cancellation – when the transmission has been aborted due to lost arbitration – when an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

Note: TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.

46.3.5.2.38 Tx Buffer Add Request Register (TXBAR)

Offset: 0x00D0 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 557. Tx Buffer Add Request Register (TXBAR)

Table 625. TXBAR field descriptions

Field	Description
0:31 AR[31:0]	Add Request Each Tx Buffer has its own Add Request bit. Writing a ‘1’ will set the corresponding Add Request bit; writing a ‘0’ has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

46.3.5.2.39 Tx Buffer Cancellation Request Register (TXBCR)

Offset: 0x00D4 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 558. Tx Buffer Cancellation Request Register (TXBCR)

Table 626. TXBCR field descriptions

Field	Description
0:31 CR[31:0]	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a ‘1’ will set the corresponding Cancellation Request bit; writing a ‘0’ has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

46.3.5.2.40 Tx Buffer Transmission Occurred Register (TXBTO)

Offset: 0x00D8 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 559. Tx Buffer Transmission Occurred Register (TXBTO)

Table 627. TXBTO field descriptions

Field	Description
0:31 TO[31:0]	Transmission Occurred Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

46.3.5.2.41 Tx Buffer Cancellation Finished Register (TXBCF)

Offset: 0x00DC Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 560. Tx Buffer Cancellation Finished Register (TXBCF)

Table 628. TXBCF field descriptions

Field	Description
0:31 CF[31:0]	Cancellation Finished Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

46.3.5.2.42 Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Offset: 0x00E0 Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 561. Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Table 629. TXBTIE field descriptions

Field	Description
0:31 TIE[31:0]	Transmission Interrupt Enable Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

46.3.5.2.43 Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Offset: 0x00E4 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 562. Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Table 630. TXBCIE field description

Field	Description
0:31 CFIE[31:0]	Cancellation Finished Interrupt Enable Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

46.3.5.2.44 Tx Event FIFO Configuration Register (TXEFC)

Offset: 0x00F0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		EFWM(1)						0		EFS(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EFSA(1)														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 563. Tx Event FIFO Configuration Register (TXEFC)

Table 631. TXEFC field descriptions

Field	Description
2:7 EFWM	Event FIFO Watermark 0x00: Watermark interrupt disabled 0x01: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) ... 0x20: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) 0x21: Watermark interrupt disabled ... 0x3F: Watermark interrupt disabled
10:15 EFS	Event FIFO Size 0x00: Tx Event FIFO disabled 0x01: Number of Tx Event FIFO elements ... 0x20: Number of Tx Event FIFO elements 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS - 1
16:29 EFSA	Event FIFO Start Address. Start address of Tx Event FIFO in Message RAM (32-bit word address,*Figure 566*.)

46.3.5.2.45 Tx Event FIFO Status Register (TXEFS)

Offset: 0x00F4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	TEFL	EFF	0	0	0	EFPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	EFGI					0	0	EFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 564. Tx Event FIFO Status Register (TXEFS)

Table 632. TXEFS field descriptions

Field	Description
6 TEFL	Tx Event FIFO Element Lost This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
7 EFF	Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
11:15 EFPI	Event FIFO Put Index Tx Event FIFO write index pointer, range 0 to 31.
19:23 EFGI	Event FIFO Get Index Tx Event FIFO read index pointer, range 0 to 31.
26:31 EFFL	Event FIFO Fill Level Number of elements stored in Tx Event FIFO, range 0 to 32.

46.3.5.2.46 Tx Event FIFO Acknowledge Register (TXEFA)

Offset: 0x00F8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	EFAI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 565. Tx Event FIFO Acknowledge Register (TXEFA)

Table 633. TXEFA field descriptions

Field	Description
27:31 EFAI	Event FIFO Acknowledge Index After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL].

46.3.6 Message RAM

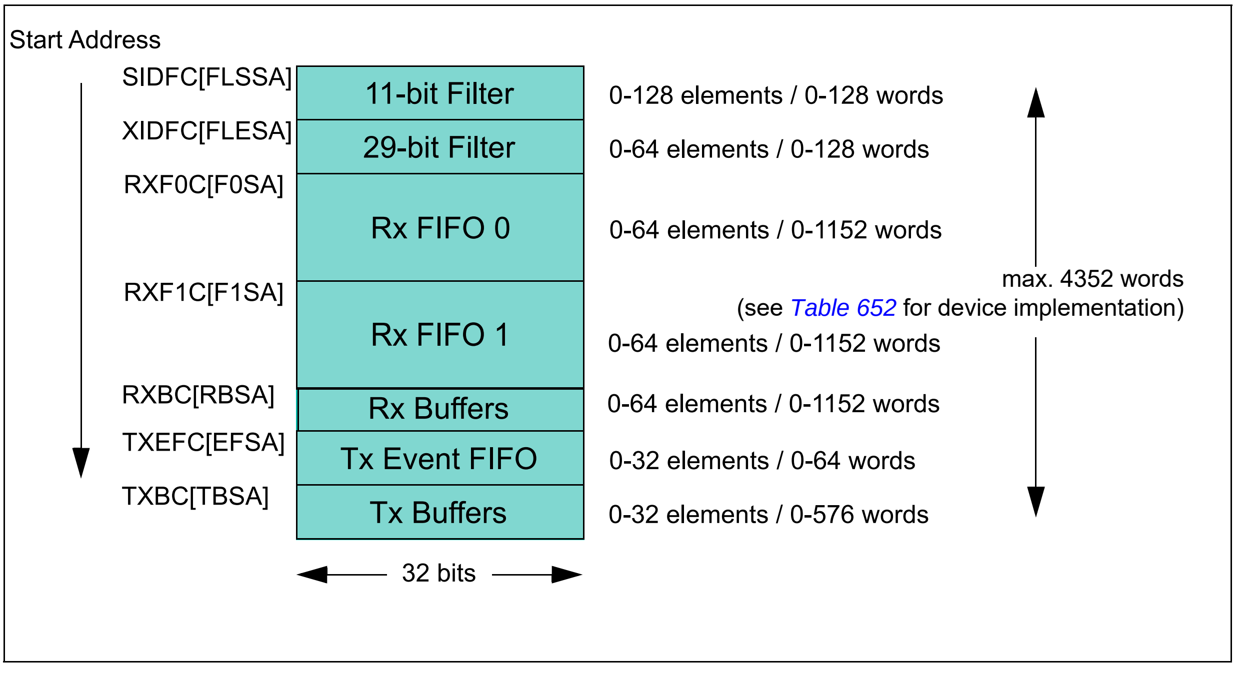
For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_CAN module.

Note: Because the Message RAM is equipped with ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing, for example, 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections will activate interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 4352 words in the Message RAM (for the chip-specific implementation details of each module's instance, refer to device configuration chapter). It is not necessary to configure each of the sections listed in *Figure 566*, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via RXESC.F0DS, RXESC.F1DS, RXESC.RBDS, and TXESC.TBDS.

Figure 566. Message RAM Configuration



When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses such as only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

46.3.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the following figure. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

	0 … 7	8 … 15	16 … 23	24 … 31
R0	ESI, XTD, RTR, ID[28:0]	ID[28:0]	ID[28:0]	ID[28:0]
R1	ANMF, FIDX[6:0]	res, EDL, BRS, DLC[3:0]	RXTS[15:0]	RXTS[15:0]
R2	DB3[7:0]	DB2[7:0]	DB1[7:0]	DB0[7:0]
R3	DB7[7:0]	DB6[7:0]	DB5[7:0]	DB4[7:0]
...	...	...	...	...
Rn	DBm[7:0]	DBm-1[7:0]	DBm-2[7:0]	DBm-3[7:0]

Figure 567. Rx Buffer and FIFO element

Table 634. Rx Buffer and FIFO element descriptions

Field	Description
R0 bit 0 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 bit 1 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit extended identifier
R0 bit 2 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame **Note:** There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = ‘1’), bit RTR reflects the state of the reserved bit r1.
R0 bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 bit 0 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element

Table 634. Rx Buffer and FIFO element descriptions (continued)

Field	Description
R1 bits 1:7 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = ‘1’). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 bit 10 EDL: Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1 bit 11 BRS: Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 bits 12:15 DLC[3:0]: Data Length Code	0-8 CAN + CAN FD: Received frame has 0-8 data bytes 9-15 CAN: Received frame has 8 data bytes 9-15 CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 bits 16:31 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 bits 0:7	DB3[7:0]: Data Byte 3
R2 bits 8:15	DB2[7:0]: Data Byte 2
R2 bits 16:23	DB1[7:0]: Data Byte 1
R2 bits 24:31	DB0[7:0]: Data Byte 0
R3 bits 0:7	DB7[7:0]: Data Byte 7
R3 bits 8:15	DB6[7:0]: Data Byte 6
R3 bits 16:23	DB5[7:0]: Data Byte 5
R3 bits 24:31	DB4[7:0]: Data Byte 4
...	...
Rn bits 0:7	DBm[7:0]: Data Byte m
Rn bits 8:15	DBm-1[7:0]: Data Byte m-1
Rn bits 16:23	DBm-2[7:0]: Data Byte m-2
Rn bits 24:31	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 3 to 17) are used for storage of a CAN message’s data field.

46.3.6.2 Tx Buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC.TFQS and TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

	0 … 7	8 … 15	16 … 23	24 … 31
T0	ESI, XTD, RTR, ID[28:0]			
T1	MM[7:0]	EFC, res, FDF, BRS, DLC[3:0]	res	
T2	DB3[7:0]	DB2[7:0]	DB1[7:0]	DB0[7:0]
T3	DB7[7:0]	DB6[7:0]	DB5[7:0]	DB4[7:0]
...	...	...	...	...
Tn	DBm[7:0]	DBm-1[7:0]	DBm-2[7:0]	DBm-3[7:0]

Figure 568. Tx Buffer element

Table 635. Tx Buffer element descriptions

Field	Description
T0 bit 0 ESI: Error State Indicator	0 ESI bit in CAN FD format depends only on error passive flag 1 ESI bit in CAN FD format transmitted recessive **Note:** The ESI bit of the transmit buffer is or'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive
T0 bit 1 XTD: Extended Identifier	0 11-bit standard identifier 1 29-bit extended identifier
T0 bit 2 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame **Note:** When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1, even if CCCR.FDOE enables the transmission in CAN FD format.
T0 bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 bits 0:7 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

Table 635. Tx Buffer element descriptions (continued)

Field	Description
T1 bit 8 EFC **Event FIFO Control**	0 Don't store Tx events 1 Store Tx events
T1 bit 10 FDF **FD Format**	0 Frame transmitted in Classic CAN format 1 Frame transmitted in CAN FD format
T1 bit 11 BRS **Bit Rat Switching**	0 CAN FD frames transmitted without bit rate switching 1 CAN FD frames transmitted with bit rate switching
T1 bits 12:15 DLC[3:0] **Data Length Code**	0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes 9-15 CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
T2 bits 0:7	DB3[7:0]: Data Byte 3
T2 bits 8:15	DB2[7:0]: Data Byte 2
T2 bits 16:23	DB1[7:0]: Data Byte 1
T2 bits 24:31	DB0[7:0]: Data Byte 0
T3 bits 0:7	DB7[7:0]: Data Byte 7
T3 bits 8:15	DB6[7:0]: Data Byte 6
T3 bits 16:23	DB5[7:0]: Data Byte 5
T3 bits 24:31	DB4[7:0]: Data Byte 4
...	...
Tn bits 0:7	DBm[7:0]: Data Byte m
Tn bits 8:15	DBm-1[7:0]: Data Byte m-1
Tn bits 16:23	DBm-2[7:0]: Data Byte m-2
Tn bits 24:31	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message's data field.

46.3.6.3 Tx Event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

	0 ... 7	8 ... 15				16 ... 23	24 ... 31
E0	ESI | XTD | RTR | ID[28:0]	ID[28:0]				ID[28:0]	ID[28:0]
E1	MM[7:0]	ET[1:0]	EDL	BRS	DLC[3:0]	TXTS[15:0]	TXTS[15:0]

Figure 569. Tx Event FIFO element

Table 636. Tx Event FIFO element descriptions

Field	Description
E0 bit 0 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 bit 1 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 bit 2 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 bits 3:31 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 bits 0:7 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.
E1 bit 8:9 ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 bit 10 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 bit 11 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching

Table 636. Tx Event FIFO element descriptions (continued)

Field	Description
E1 bits 12:15 DLC[3:0]	Data Length Code 0-8 CAN + CAN FD: frame with 0-8 data bytes transmitted 9-15 CAN: frame with 8 data bytes transmitted 9-15 CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
E1 bits 16:31 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

46.3.6.4 Standard Message ID Filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC[FLSSA] plus the index of the filter element (0 to 127).

	0		7 | 8	15 | 16	23 | 24 31
S0	SFT[1:0]	SFEC[2:0]	SFID1[10:0]	res	SFID2[10:0]

Figure 570. Standard Message ID Filter element

Table 637. Standard Message ID Filter element descriptions

Field	Description
Bits 0:1 SFT[1:0]	Standard Filter Type 00 Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1) 01 Dual ID filter for SFID1 or SFID2 10 Classic filter: SFID1 = filter, SFID2 = mask 11 Reserved
Bit 2:4 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = ‘100’, ‘101’, or ‘110’ a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

Table 637. Standard Message ID Filter element descriptions (continued)

Field	Description
Bits 5:15 SFID1[10:0]	Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 21:31 SFID2[10:0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: 1) SFEC = ‘001’...‘110’ Second ID of standard ID filter element 2) SFEC = ‘111’ Filter for Rx Buffers or for debug messages
	SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C
	SFID2[8:6] is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_CAN Host clock period in case the filter matches.
	SFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

46.3.6.5 Extended Message ID Filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0 to 63).

	0		7 | 8 … 15 | 16 … 23 | 24 … 31
F0	EFEC[2:0]		EFID1[28:0]
F1	EFT[1:0]	res	EFID2[28:0]

Figure 571. Extended Message ID Filter element

Table 638. Extended Message ID Filter element description

Field	Description
F0 bits 0:2 EFEC[2:0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = ‘100’, ‘101’, or ‘110’ a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored
F0 bits 3:31 EFID1[28:0]	Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (refer to *Section 46.3.10.1.5*) is used.
F1 bits 0:1 EFT[1:0]	Extended Filter Type 00 Range filter from EFID1 to EFID1 (EFID2 ≥ EFID1) 01 Dual ID filter for EFID1 or EFID2 10 Classic filter: EFID1 = filter, EFID2 = mask 11 Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), XIDAM mask not applied
F1 bits 3:31 EFID2[28:0]	This bit field has a different meaning depending on the configuration of EFEC: 1) EFEC = ‘001’...‘110’ Second ID of extended ID filter element 2) EFEC = ‘111’ Filter for Rx Buffers or for debug messages EFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C EFID2[8:6] is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one Host clock period in case the filter matches. EFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

46.3.7 M_CAN functional description

46.3.7.1 Operating modes

46.3.7.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus transmit output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = ‘1’. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set

- HPMS–High Priority Message Status
- RXF0S–Rx FIFO 0 Status
- RXF1S–Rx FIFO 1 Status
- TXFQS–Tx FIFO/Queue Status
- TXBRP–Tx Buffer Request Pending
- TXBTO–Tx Buffer Transmission Occurred
- TXBCF–Tx Buffer Cancellation Finished
- TXEFS–Tx Event FIFO Status

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = ‘1’.

The following registers are only writable while CCCR[CCE] = ‘0’

- TXBAR–Tx Buffer Add Request
- TXBCR–Tx Buffer Cancellation Request

CCCR[TEST] and CCCR.MON can only be set by the Host while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’. Both bits may be reset at any time. CCCR.DAR can only be set/reset while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’.

The following sequence can be used as an example code for initializing the CAN module (M_CAN0 in example):

```
-- Setting the INIT bit high to start the initialization
      while (read_data != 0x003){
         write(M_CAN0,0xF,driver.regs.can0_cccr.addr,0x003);
            read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
      };
```

```
-- Configuring the time stamp and timeout registers
     write(M_CAN0,0xF,driver.regs.can0_tscc.addr,0x000);
     write(M_CAN0,0xF,driver.regs.can0_tocc.addr,0x000);

-- Configuring the interrupt registers
     write(M_CAN0,0xF,driver.regs.can0_ie.addr,0xFFFFFFFF);
     write(M_CAN0,0xF,driver.regs.can0_ils.addr,0x00000002);
     write(M_CAN0,0xF,driver.regs.can0_ile.addr,0x00000003);

     write(M_CAN0,0xF,driver.regs.can0_gfc.addr,0x000);
     write(M_CAN0,0xF,driver.regs.can0_xidam.addr,0x00FFFFFF);

-- Writing the start addresses of all the M_CAN Buffers
     generate and drive M_CAN0 can0_rxf0c keeping {
                                         .f0sa == 0x100;
                                         .f0s  == 0x03;
                                         .f0wm == 0x01;
                                    };
     generate and drive M_CAN0 can0_rxf1c keeping {
                                           .f1sa == 0x200;
                                           .f1s  == 0x03;
                                           .f1wm == 0x01;
                                           };
     generate and drive M_CAN0 can0_sidfc keeping {
                                           .flssa == 0x000;
                                           .lss   == 0x80;
                                           };
     generate and drive M_CAN0 can0_xidfc keeping {
                                           .flesa == 0x080;
                                           .lse   == 0x40;
                                           };
     generate and drive M_CAN0 can0_txbc  keeping {
                                           .tbsa == 0x300;
                                           .ndtb == 0x020;
                                           .tfqs == 0x1D;
                                           };
     generate and drive M_CAN0 can0_txefc keeping {
                                         .efsa == 0x380;
                                         .efs == 0x020;
                                         .efwm == 0x00;
                                         };
-- Writing to the RAM for filter configuration
===================================================
-- Writing to TX Buffer
```

```
        write(RAM,0xF,0xC00,0x04040000);
        write(RAM,0xF,0xC04,0x10830000);
        write(RAM,0xF,0xC08,0x00010001);
        write(RAM,0xF,0xC10,0x50000002);
        write(RAM,0xF,0xC14,0x11830000);
        write(RAM,0xF,0xC18,0x00070102);

-- Writing to s-filter
        write(RAM,0xF,0x000,0x58020004);
        write(RAM,0xF,0x004,0x08000005);
        write(RAM,0xF,0x008,0x58060007);
        write(RAM,0xF,0x00C,0x08030008);
        write(RAM,0xF,0x010,0x880E07FB);
        write(RAM,0xF,0x014,0x980D07FE);
        write(RAM,0xF,0x018,0x68100100);
        write(RAM,0xF,0x01C,0x70120200);
        write(RAM,0xF,0x020,0x981F07F6);
        write(RAM,0xF,0x024,0x50140014);
        write(RAM,0xF,0x028,0x58160017);
        write(RAM,0xF,0x02C,0x08110019);
        write(RAM,0xF,0x030,0x4A22001A);
        write(RAM,0xF,0x034,0xA01A07FF);
        write(RAM,0xF,0x038,0x5BFF07E3);
        write(RAM,0xF,0x03C,0x080007FF);

-- Writing to x-filter
        write(RAM,0xF,0x200,0x201ABCD9);
        write(RAM,0xF,0x204,0x001ABCE5);
        write(RAM,0xF,0x208,0x403BCDEF);
        write(RAM,0xF,0x20C,0x003BCDEF);
        write(RAM,0xF,0x210,0x605CDEF2);
        write(RAM,0xF,0x214,0x405CDEF0);
        write(RAM,0xF,0x218,0x407DEEFF);
        write(RAM,0xF,0x21C,0x007DEF01);
        write(RAM,0xF,0x220,0x209ABCDE);
        write(RAM,0xF,0x224,0x009ABCDF);
        write(RAM,0xF,0x228,0xA0BBCDEF);
        write(RAM,0xF,0x22C,0x40BBCDEE);
        write(RAM,0xF,0x230,0x20DCDEF0);
        write(RAM,0xF,0x234,0x9FFFFFFF);
        write(RAM,0xF,0x238,0x40000000);
        write(RAM,0xF,0x23C,0x1FFFFFFF);
==================================================
write(M_CAN0,0xF,driver.regs.can0_nbtp.addr,0x00002303);
```

```
print "Configuration done";
read_data = 0x01;
while (read_data != 0x00){
  write(M_CAN0,0xF,driver.regs.can0_cccr.addr, 0x000);
    wait [1];
    read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
  };

print "Initialization done";
```

46.3.7.2 Normal operation

Once the M_CAN is initialized and CCCR[INIT] is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, the received messages including Message ID and DLC are stored into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers or a Tx FIFO, or both, or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

46.3.7.3 CAN FD Operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = *recessive* signifies a CAN FD frame, FDF = *dominant* signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = *dominant* and BRS = *recessive*. The coding of res = *recessive* is reserved for future expansion of the protocol. In case the M_CAN receives a frame with FDF = *recessive* and res = *recessive*, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled (CCCR.PXHD = ‘0’), this causes the operation state to change from Receiver (PSR.ACT = “10”) to Integrating (PSR.ACT = “00”) at the next sample point. In case Protocol Exception Handling is disabled (CCCR.PXHD = ‘1’), the M_CAN will treat a *recessive* res bit as a form error and will respond with an error frame.

CAN FD operation is enabled by programming CCCR.FDOE. In case CCCR.FDOE = ‘1’, transmission and reception of CAN FD frames are enabled. Transmission and reception of Classic CAN frames are always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With CCCR.FDOE = ‘0’, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set.

With CCCR.FDOE = ‘0’, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With CCCR.FDOE = ‘1’ and CCCR.BRSE = ‘0’, only bit FDF of a Tx Buffer element is evaluated. With CCCR.FDOE = ‘1’ and CCCR.BRSE = ‘1’, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation
- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to *Table 639*.

Table 639. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is *recessive*. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the Nominal Bit Timing and Prescaler Register NBTP. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing and Prescaler Register DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (CAN clock). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the bit ESI (Error Status Indicator) is determined by the transmitter’s error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted *recessive*, else it is transmitted *dominant*.

46.3.7.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via CAN Tx output pin the M_CAN the transmitted data from its local CAN transceiver via CAN Rx input pin. The received data is delayed by the transmitter delay. In case this delay is greater than NTSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

46.3.7.4.1 Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration the transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP.TDC.phase the delay compensation is always disabled.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the M_CAN's Tx output through the transceiver to the CAN Rx input plus the transmitter delay compensation offset as configured by TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

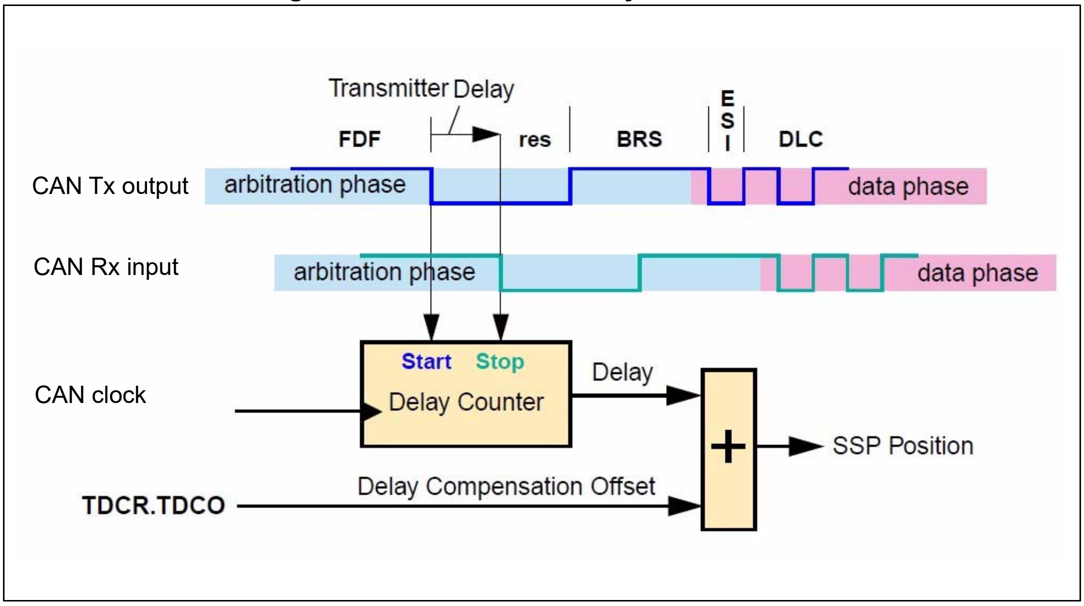
The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_CAN:

- The sum of the measured delay from CAN Tx output to CAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less than 6 bit times in the data phase
- The sum of the measured delay from CAN Tx output to CAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less than or equal to 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs

Figure 572 below describes how the Transceiver Delay Compensation measurement.

If transmitter delay compensation is enabled by programming DBTP.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input CAN Rx input of the transmitter. The resolution of this measurement is one mtq.

Figure 572. Transceiver Delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on CAN Rx input, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCR.TDCF AND CAN Rx input is low.

46.3.7.5 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The Host can set the M_CAN into Restricted Operation mode by setting bit CCCR.ASM. The bit can only be set by the Host when both CCCR.CCE and CCCR.INIT are set to ‘1’. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

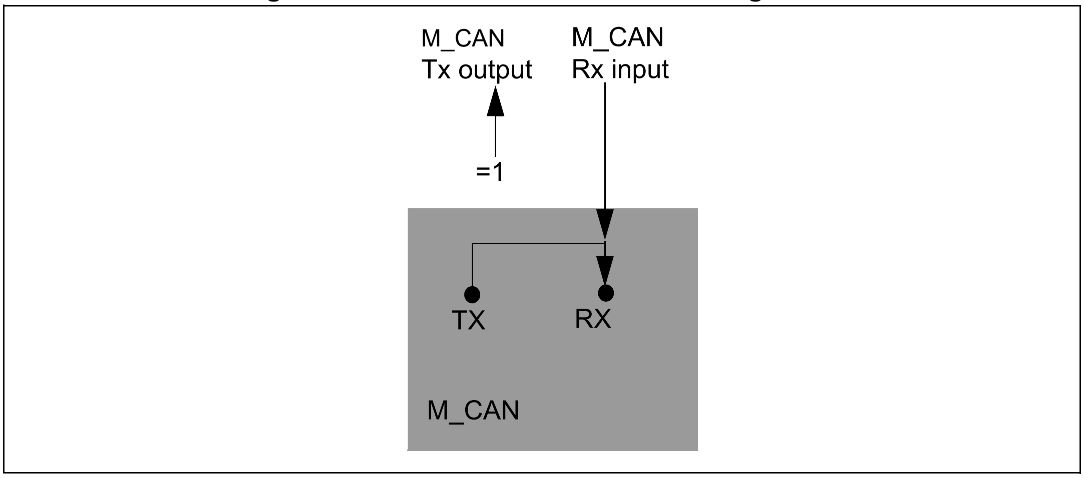
If the M_CAN is connected to a Clock Calibration on CAN Unit, CCCR.ASM is controlled by “calibration OK” input. In case “calibration OK” switches to ‘0’, bit CCCR.ASM is set. When “calibration OK” switches back to ‘1’, bit CCCR.ASM returns to the previously written value. The state of CCCR.ASM is the written value while “calibration OK” input is at ‘1’. The input is hardwired to ‘1’ when there is no Clock Calibration on CAN Unit connected.

46.3.7.6 Bus monitoring mode

The M_CAN is set in Bus Monitoring Mode by programming CCCR[MON] to one. In Bus Monitoring Mode (refer to ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode, register TXBRP is held in reset state. The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.

Figure 573 shows the connection of M_CAN Tx and Rx signals to the M_CAN in Bus Monitoring Mode.

Figure 573. Pin control in Bus Monitoring mode



46.3.7.7 Disabled automatic retransmission

According to the CAN Specification (refer to ISO11898-1, 6.3.3 Recovery Management), the M_CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

46.3.7.7.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP[TRPx] is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = ‘10’ (transmission in spite of cancellation).

46.3.7.8 Power down (Sleep mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as one. When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module clock inputs (CAN clock and Host clock) may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting signal clock stop request signal resp. CC Control Register flag CCCR[CSR]. The M_CAN will acknowledge this by resetting clock stop acknowledge output signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

46.3.7.9 Test modes

To enable write access to register TEST (refer to *Section 46.3.5.2.4: Test Register (TEST)*), bit CCCR[TEST] has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN Tx output pin by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN’s bit timing and it can drive constant dominant or recessive values. The actual value at M_CAN Rx pin can be read from TEST[RX]. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST[TX] until the new configuration is visible at output Tx pin. This applies also when reading input Rx pin via TEST[RX].

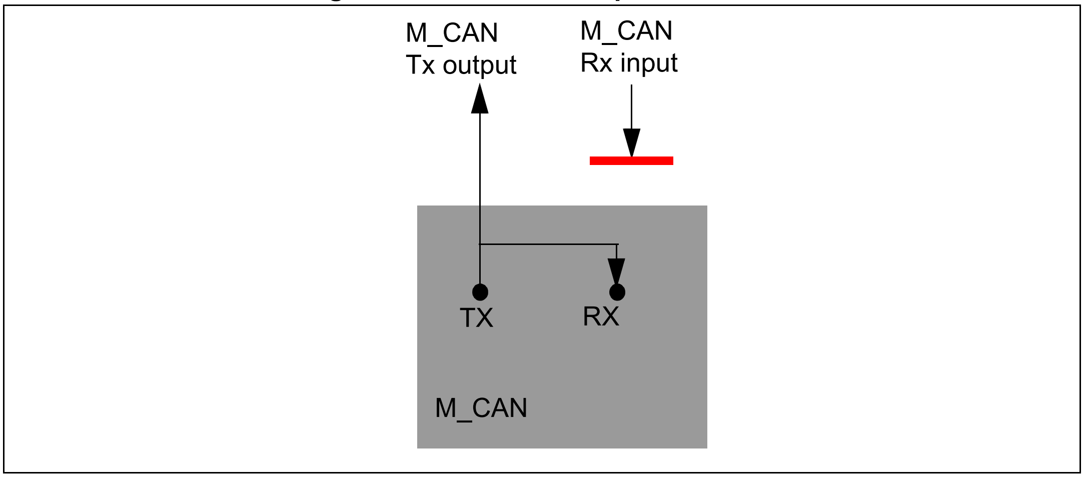
Note: Test modes should be used for production tests or self test only. The software control for Tx pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

46.3.7.9.1 External Loop Back mode

The M_CAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 574* shows the connection of Tx and Rx signals to the M_CAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN Rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the CAN Tx output pin.

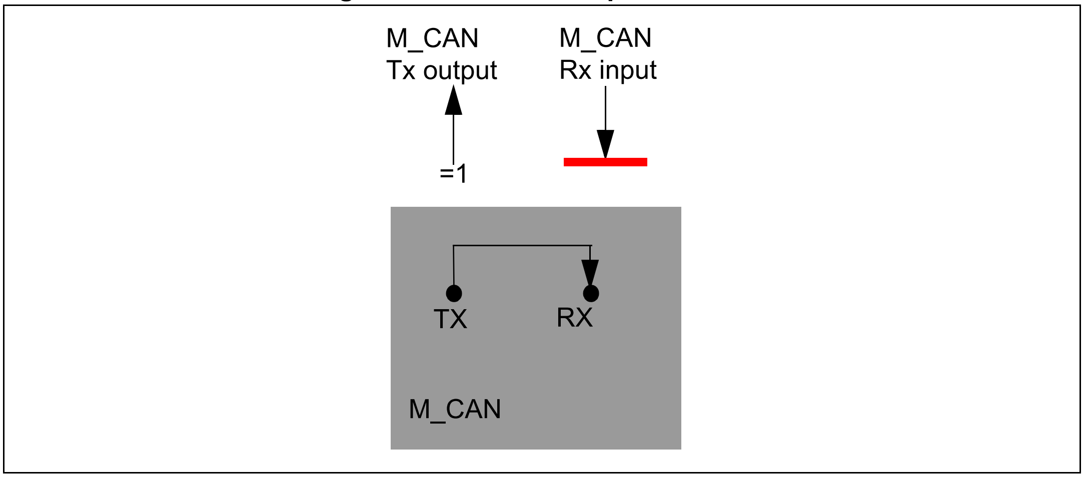
Figure 574. External Loop Back mode



46.3.7.9.2 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST[LBCK] and CCCR[MON] to one. This mode can be used for a “Hot Selftest”, meaning the M_CAN can be tested without affecting a running CAN system connected to the CAN Tx output and Rx input pins. In this mode M_CAN Tx pin is disconnected from the M_CAN and M_CAN Tx pin is held recessive. *Figure 575* shows the connection of M_CAN Tx pin and Rx to the M_CAN in case of Internal Loop Back mode.

Figure 575. Internal loop back mode



46.3.8 Timestamp generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler TSCC[TCP] can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via TSCV[TSC]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer /Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.

46.3.9 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The Timeout Counter can only be started while CCCR[INIT] = ‘0’. It is stopped when CCCR[INIT] = ‘1’, for example when the M_CAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: The clock signal for the Timeout Counter is derived from the CAN core’s sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

46.3.10 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indexes.

46.3.10.1 Acceptance filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (such as CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.DLEC.

Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Section 46.3.10.2.2: Rx FIFO Overwrite Mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into a Rx Buffer, the unmodified received identifier is stored independently of the filter(s) used. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

46.3.10.1.1 Range filter

The filter matches for all received frames with Message IDs in the range defined by SFID1/SFID2 resp. EFID1/EFID2. There are two possibilities when range filtering is used together with extended frames:

- EFT = ‘00’: The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = ‘11’: The Extended ID AND Mask (XIDAM) is not used for range filtering

46.3.10.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SFID1 = SFID2 resp. EFID1 =EFID2.

46.3.10.1.3 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SFID1/EFID2 is used as Message ID filter, while SFID2/EFID2 is used as filter mask.

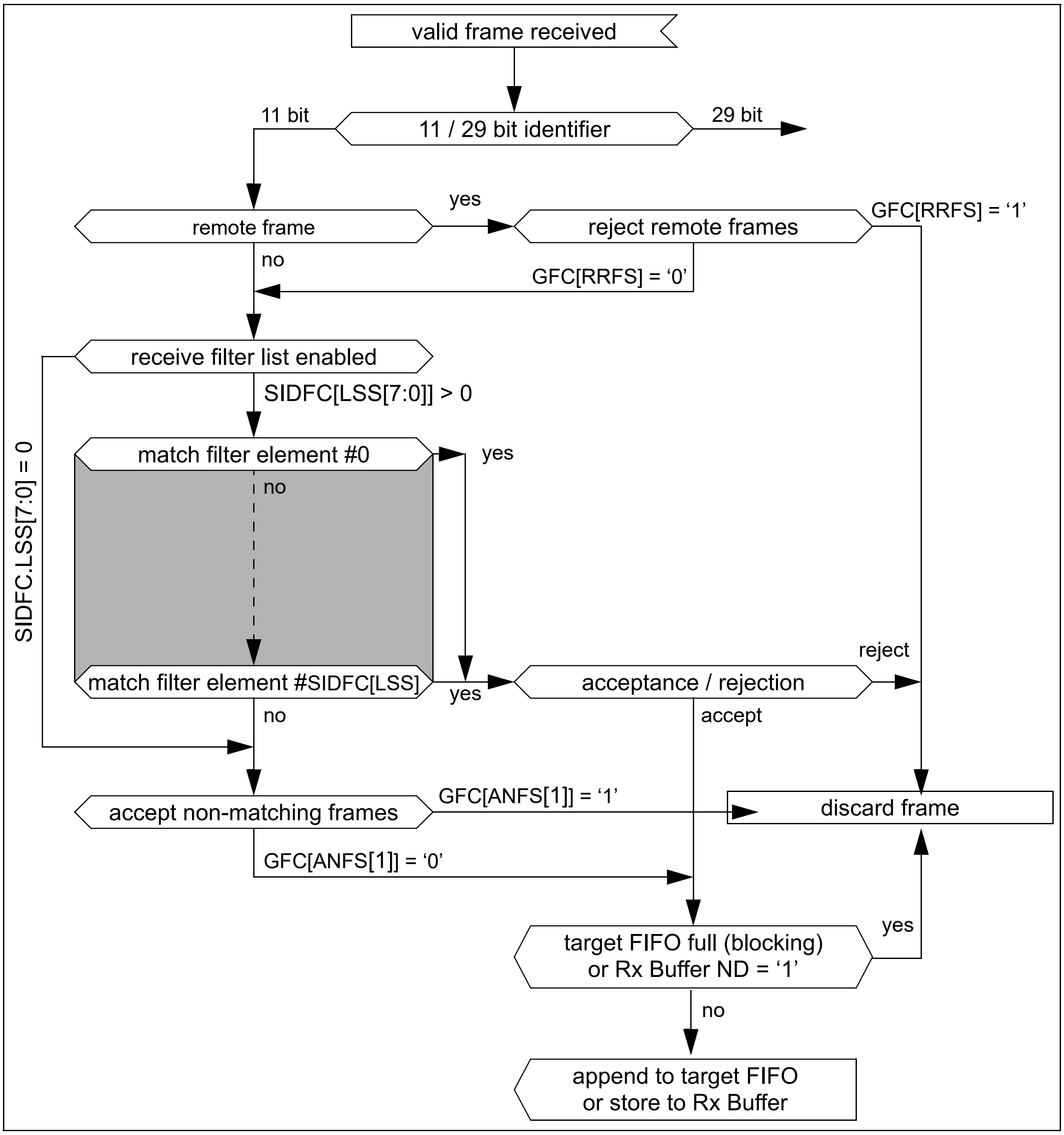
A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, for example the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering. In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

46.3.10.1.4 Standard message ID filtering

Figure 576 shows the flow for standard message ID (11-bit Identifier) filtering.

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 576. Standard Message ID Filter Path



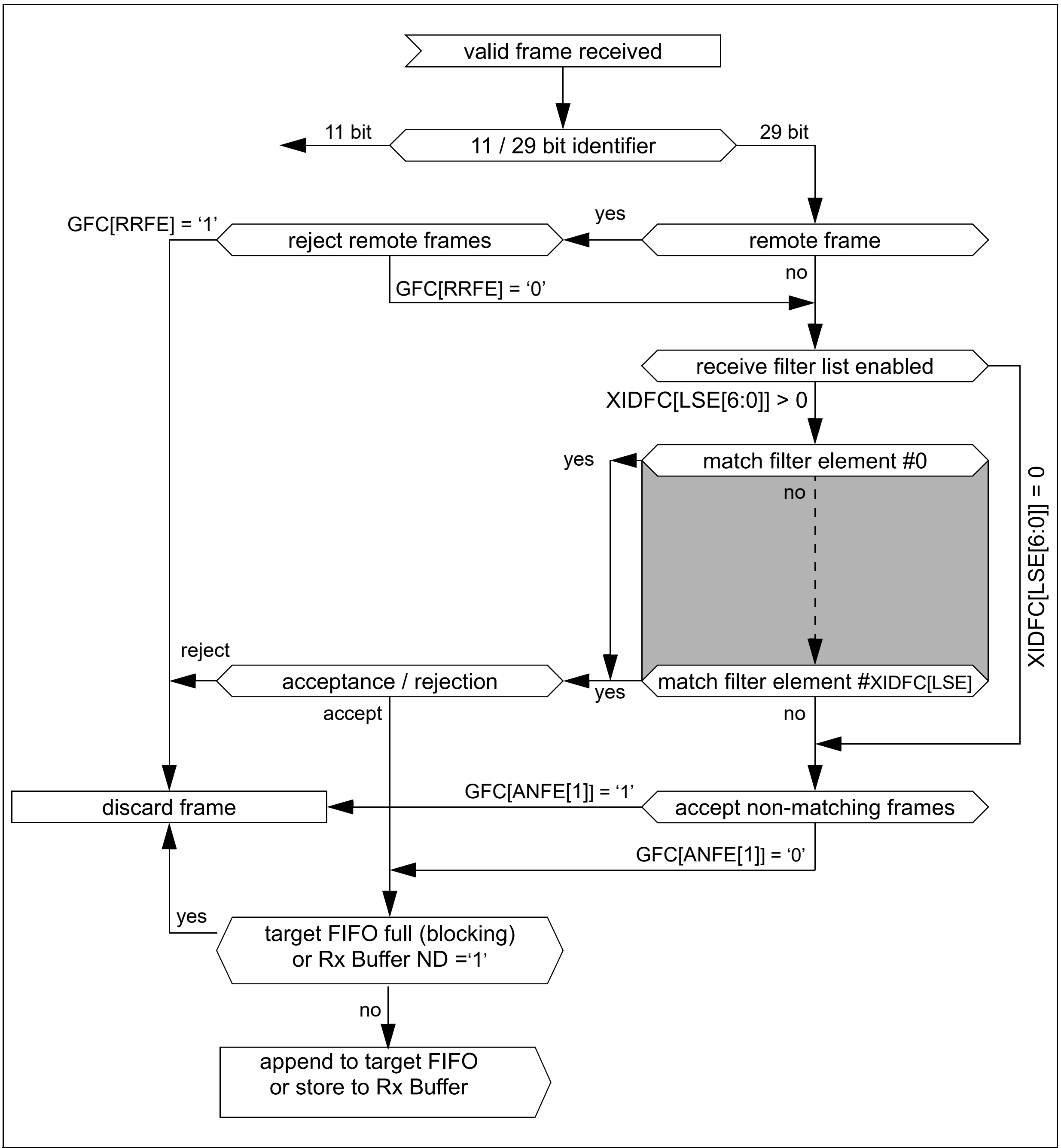
46.3.10.1.5 Extended message ID filtering

Figure 577 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in *Section 46.3.5.2.21: Extended ID Filter Configuration Register (XIDFC)*.

Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements which is controlled by the

Global Filter Configuration (GFC) and the Extended ID Filter Configuration (XIDFC) Message ID. The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

Figure 577. Extended Message ID Filter Path



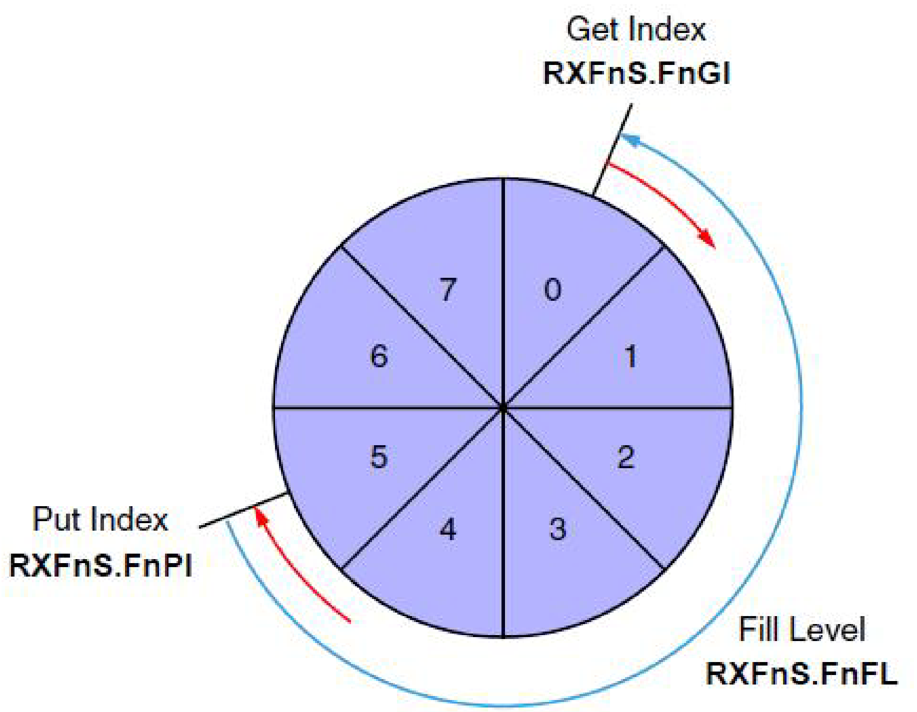
46.3.10.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 refer to *Section 46.3.10.1: Acceptance filtering*. The Rx FIFO element is described in *Section 46.3.6.1: Rx Buffer and FIFO element*.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag IR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag IR.RFnF is set.

Figure 578. Rx FIFO Status



When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI × FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 640. Rx Buffer / FIFO Element Size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10

Table 640. Rx Buffer / FIFO Element Size (continued)

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
110	48	14
111	64	18

46.3.10.2.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = ‘0’. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = ‘1’. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = ‘1’. In addition interrupt flag IR.RFnL is set.

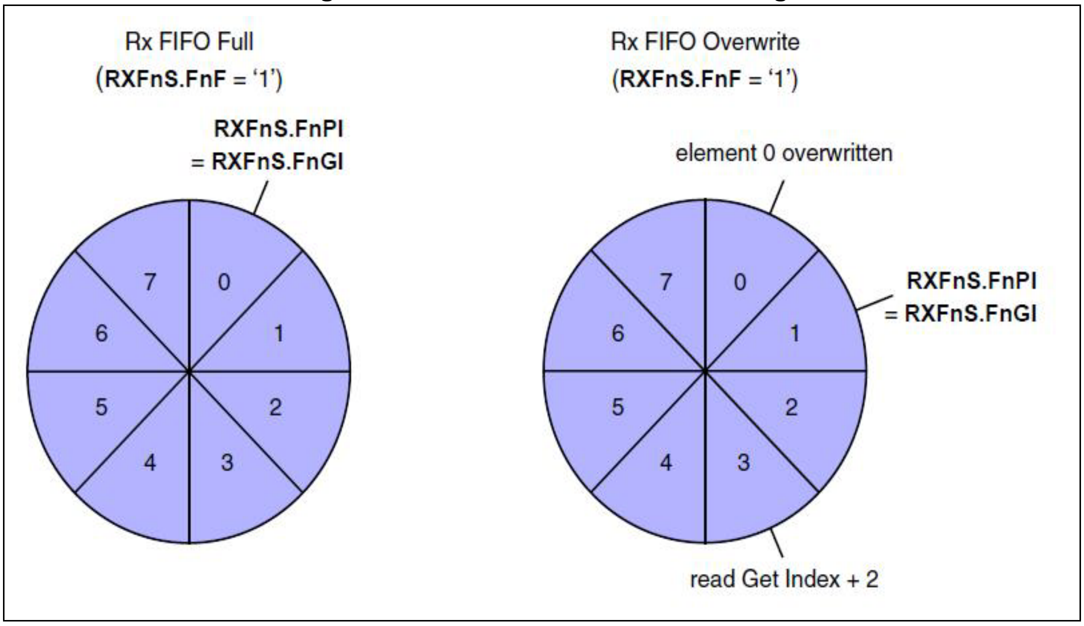
46.3.10.2.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = ‘1’.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = ‘1’, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. *Figure 579* shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Figure 579. Rx FIFO Overflow Handling



After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = '0').

46.3.11 Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = '111' and SFID2 / EFID2[10:9] = '00' has to be configured (refer to *Section 46.3.6.4: Standard Message ID Filter element* and *Section 46.3.6.5: Extended Message ID Filter element*).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 641. Example Filter Configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message 1	00	00 0000
1	ID debug message 2	00	00 0001
2	ID debug message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

46.3.11.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

46.3.12 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (for example #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (refer to *Section 46.3.6.1: Rx Buffer and FIFO element*).

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = ‘111’ have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while DMA request output is activated. The behavior is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets the DMA acknowledge input . This resets the DMA request output. Now the M_CAN is prepared to receive the next set of debug messages.

46.3.12.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to ‘111’. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 /EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

Table 642. Example Filter Configuration for debug messages

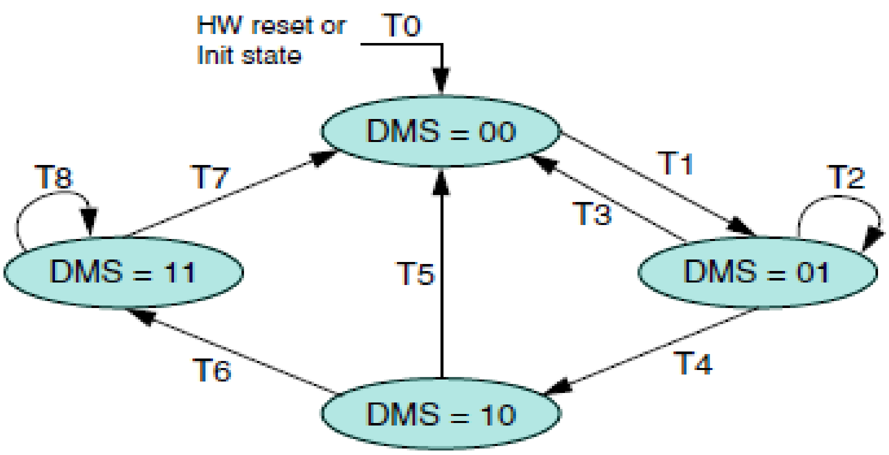
Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

46.3.12.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S.DMS.

Figure 580. Debug Message Handling State Machine



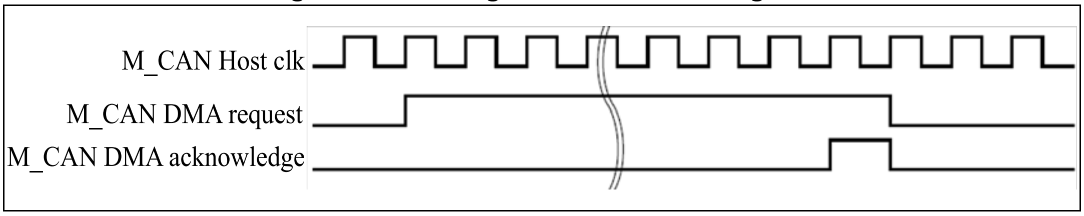
T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C
T1: reception of debug message A
T2: reception of debug message A
T3: reception of debug message C
T4: reception of debug message B
T5: reception of debug messages A, B
T6: reception of debug message C
T7: DMA transfer completed
T8: reception of debug message A,B,C (message rejected)

46.3.12.3 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_CAN DMA request signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_CAN while M_CAN DMA request signal is active.

After the transfer of the received messages has completed the DMA unit activates the M_CAN DMA ACK signal. This resets M_CAN DMA request signal. The debug message handling state machine enters idle state (DMS = ‘00’) and waits for reception of the next debug messages.

Figure 581. Timing of DMA interface signals



Note: If the DMA unit activates input M_CAN DMA acknowledge signal before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

46.3.13 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indexes, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in the Note: on page 1017.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

46.3.13.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU’s CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If for example CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR.TXP. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR.TXP = ‘0’).

This feature looses up burst transmissions coming from a single node and it protects against “babbling idiot” scenarios where the application program erroneously requests too many transmissions.

46.3.13.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (refer to *Table 643*). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

Table 643. Tx Buffer / FIFO / Queue Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

46.3.13.3 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to ‘0’. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF] = ‘1’) is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a ‘1’ to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFOs Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (refer to *Table 643*). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

46.3.13.4 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to ‘1’. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first. New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF] = ‘1’), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

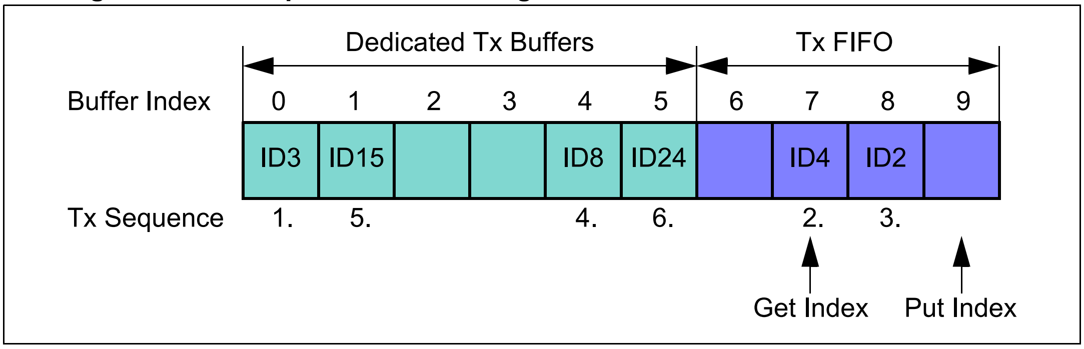
The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (refer to *Table 643*). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

46.3.13.5 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 582. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO



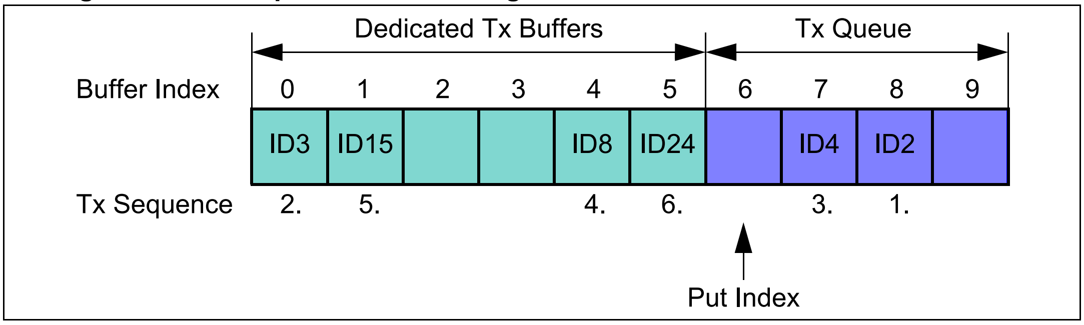
Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS[TFGI])
- Buffer with lowest Message ID gets highest priority and is transmitted next

46.3.13.6 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Queue Buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 583. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue



Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

46.3.13.7 Transmit cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a ‘1’ to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to ‘1’.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

46.3.13.8 Tx Event handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in the *Note: on page 1019*. When a Tx Event FIFO full condition is signaled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

46.3.14 FIFO acknowledge handling

The Get Indexes of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (refer to *Section 46.3.5.2.28: Rx FIFO 0 Acknowledge Register (RXF0A)* and *Section 46.3.5.2.46: Tx Event FIFO Acknowledge Register (TXEFA)*). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFOs Get Index.

Due to the fact that the CPU has free access to the M_CAN’s Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFOs Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFOs Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

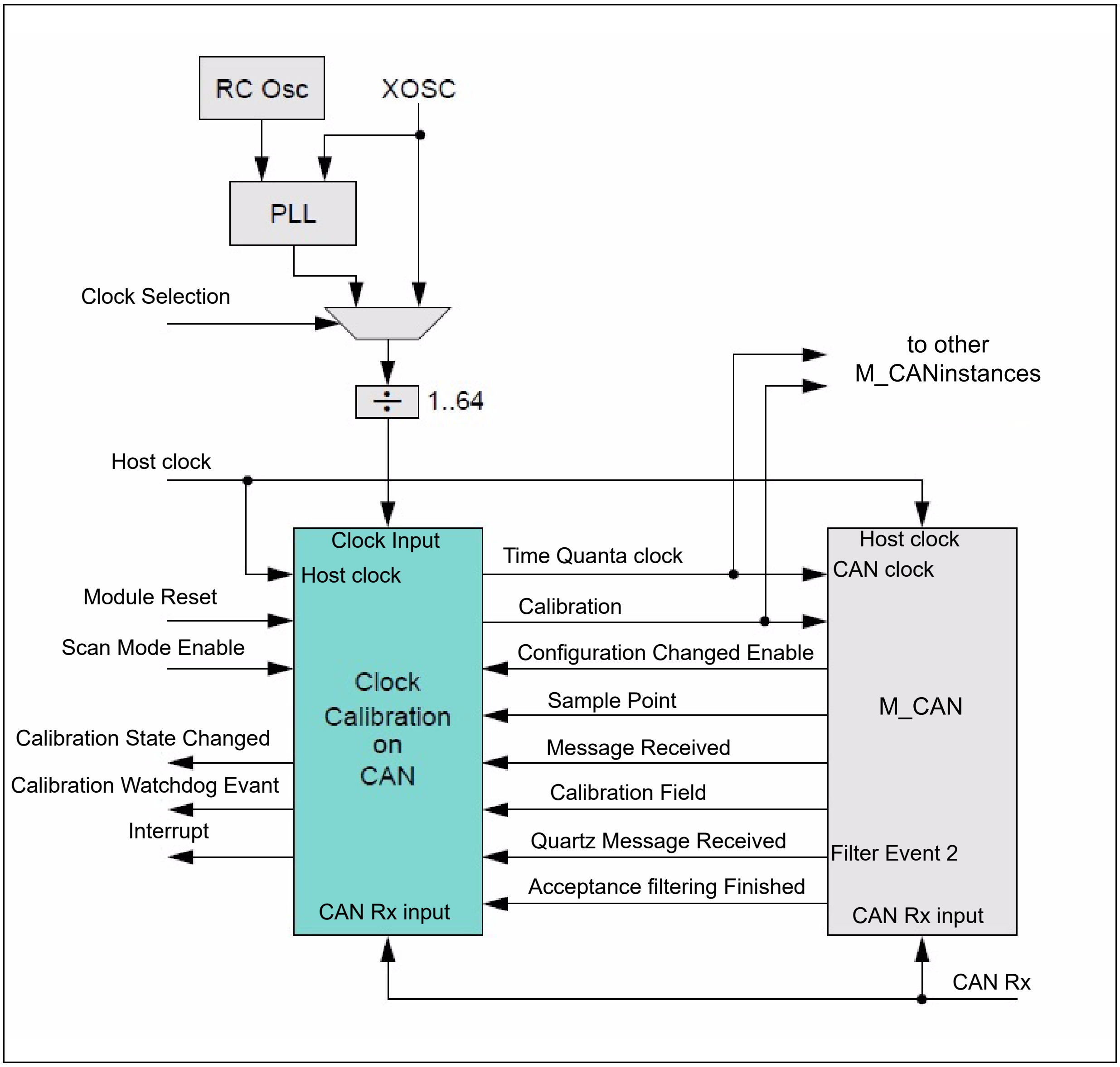
46.3.15 Clock Calibration on CAN Unit (CCCU)

This section describes an add-on for the M_CAN modules that generates a calibrated CAN Time Quanta Clock from an on-chip RC-oscillator clock by evaluating CAN messages received by the attached M_CAN. Calibration on CAN is only possible when the M_CAN is operated in standard CAN mode.

Note: If calibration unit is intended to generate the protocol clock, then it is mandatory to configure calibration unit and MCAN1 before use the other CAN controller like MCAN2 and MCAN3, because the calibrated clock is generated through MCAN1.

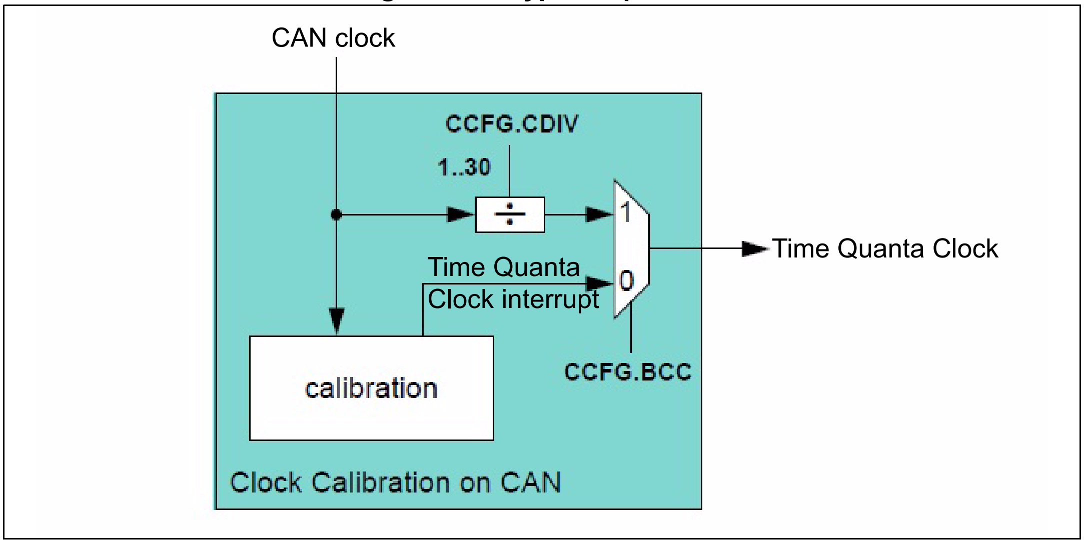
Figure 584 shows the interface of the Clock Calibration on CAN Unit. The clock multiplexer controlled by clock input from Clock Select multiplexer controls whether the clock from the PLL or XOSC is routed to Clock Select multiplexer input. The calibrated Time Quanta Clock is routed to all CAN instances. In case of a successful calibration this is signaled via Calibration.

Figure 584. Connection of Clock Calibration on CAN Unit to M_CAN



CPU accesses to the Clock Calibration on CAN Unit use the Host clock connected to Host clock input from Clock Select Multiplexer. Clock calibration is bypassed when CCFG.BCC = ‘1’. In this case, the clock from Clock Select multiplexer is routed through a clock divider to Time Quanta Clock output (refer to *Figure 585*).

Figure 585. Bypass operation



46.3.15.1 Operating Conditions

The Clock Calibration on CAN Unit is designed to operate under the following conditions:

- Clock input from PLL (clock input from Clock Select multiplexer) between 80 MHz and 500 MHz
- M_CAN bit rates between 125 kbit/s and 1 Mbit/s
- Clock for Host accesses (Host clock) is 40/50 MHz

The Clock Calibration on CAN Unit generates a calibrated Time Quanta clock in the range from 0.5 MHz to 25 MHz.

Note: The M_CAN requires that the CAN clock is always below or equal to the Host clock (CAN clock ≤ Host clock). This has to be considered when the Clock Calibration on CAN Unit is bypassed (CCFG.BCC = '1').

46.3.15.2 Calibration Accuracy

The calibration accuracy in state Precision_Calibrated depends on:

- The dynamic clock tolerance at the clock input from clock select multiplexer
- The measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one clock input from Clock Select multiplexer period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages has to be chosen to fit the clock tolerance requirements of the attached M_CANs.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages, such as, caused by change of temperature or operating voltage.

46.3.15.3 Memory map and register description

46.3.15.3.1 Hardware Reset Description

After hardware reset, the registers of the Clock Calibration on CAN Unit hold the reset values listed in *Table 644*.

The Clock Calibration on CAN Unit allocates an address space of 16 bytes. All registers are organized as 32-bit registers. The Clock Calibration on CAN Unit is accessible by the Host CPU using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access). Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when Configuration Change Enable input = ‘1’. Configuration Change Enable signal is activated when the M_CAN control bits CCCR.CCE = ’1’ AND CCCR.INIT = ‘1’.

Table 644. CCCU memory map

Address Offset	Register Name	Access(1)	Reset value	Location
0x0000	Core Release Register (CREL)	R	0x1114_0729	*Section 46.3.15.3.2*
0x0004	Calibration Configuration Register (CCFG)	RP	0x0000_0404	*Section 46.3.15.3.3*
0x0008	Calibration Status Register (CSTAT)	R	0x0203_FFFF	*Section 46.3.15.3.4*
0x000C	Calibration Watchdog (CWD)	RP	0x0000_0000	*Section 46.3.15.3.5*
0x0010	CU Interrupt Register (CUIR)	R/W	0x0000_0000	*Section 46.3.15.3.6*
0x0014	CU Interrupt Enable (CUIE)	R/W	0x0000_0000	*Section 46.3.15.3.7*

1. R = Read, W = Write, P = Protected Write

46.3.15.3.2 Core Release Register (CREL)

Offset: 0x0000 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset(1)	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-d	-d	-d	-d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset(1)	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d

1. -r = release, -d = time stamp, the coding of revisions depends on the module version used in the device. For the device reset value refer to *Table 644: CCCU memory map*.

Figure 586. Core Release Register (CREL)

Table 645. CREL field descriptions

Field	Description
0:3 REL	Core Release One digit, BCD-coded.
4:7 STEP	Step of Core Release One digit, BCD-coded.
8:11 SUBSTEP	Substep of Core Release One digit, BCD-coded.
12:15 YEAR	Time Stamp Year One digit, BCD-coded.
16:23 MON	Time Stamp Month Two digits, BCD-coded.
24:31 DAY	Time Stamp Day Two digits, BCD-coded.

Table 646. Example for Coding of Revisions

Release	Step	SubStep	Year	Month	Day	Name
1	0	0	2	02	20	Revision 1.0.0, Date 2012/02/20

46.3.15.3.3 Calibration Configuration Register (CCFG)

Offset: 0x0004 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWR	0	0	0	0	0	0	0	0	0	0	0	CDIV(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OCPM(1)								CFL(1)	BCC(1)	0	TQBT(1)				
W																
Reset	0	0	0	0	0	1	0	0	0	0/1	0	0	0	1	0	0

1. Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when Configuration Change Enable input = ‘1’. Configuration Change Enable signal is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Figure 587. Calibration Configuration Register (CCFG)

Table 647. CCFG field descriptions

Field	Description
0 SWR	Software Reset Writing a ‘1’ to this bit will reset the calibration FSM to state Not_Calibrated (CSTAT.CALS = ‘00’). The Calibration Watchdog value CWD.WDV is also reset. Registers CCFG, CSTAT and the Calibration Watchdog configuration CWD.WDC are unchanged. The bit remains set until reset has completed.
12:15 CDIV	Clock Divider The clock divider has to be configured when the clock calibration is bypassed (BCC = ‘1’) to assure that the M_CAN requirement CAN clock <= Host clock is fulfilled. 0000 Divide by 1 0001 Divide by 2 0010 Divide by 4 0011 Divide by 6 0100 Divide by 8 0101 Divide by 10 0110 Divide by 12 0111 Divide by 14 1000 Divide by 16 1001 Divide by 18 1010 Divide by 20 1011 Divide by 22 1100 Divide by 24 1101 Divide by 26 1110 Divide by 28 1111 Divide by 30
16:23 OCPM	Oscillator Clock Periods Minimum (0x00–0xFF) Configures the minimum number of clock input from Clock Select multiplexer periods in two CAN bit times. OCPM is used in Basic Calibration to avoid false measurements in case of glitches on the bus line. The configured number of clock input from Clock Select multiplexer periods is OCPM $\times$ 32. The configuration depends on the clock input from Clock Select multiplexer frequency (80 MHz to 500 MHz) and the bit rate configured in the attached M_CANs (125 kbit/s to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1.6 bit times at 80 MHz clock input from Clock Select multiplexer and 1 Mbit/s CAN bit rate.
24 CFL	Calibration Field Length 0 Calibragation field length is 32 bits. 1 Calibragation field length is 64 bits.

Table 647. CCFG field descriptions (continued)

Field	Description
25 BCC	Bypass Clock Calibration If this bit is set, the clock input clock input from Clock Select multiplexer is routed to the Time Quanta clock output through a clock divider configurable via CDIV, Calibration is always ‘1’. In this case the baud rate prescaler of the connected M_CANs has to be configured to generate the M_CAN internal time quanta clock. 0 Clock calibration unit generates Time Quanta clock 1 Clock calibration unit bypassed **Note:** As long as clock input from Clock Select multiplexer is equal or above 80 MHz the Clock Calibration on CAN Unit is functional, even when BCC = ‘1’. The calibration state can be read from register CSTAT.
27:31 TQBT	Time Quanta per Bit Time (0x04–0x19) Configures the number of time quanta per bit time. Same value as configured in the attached M_CANs. The range of the resulting Time Quanta clock is from 0,5 MHz (bit rate of 125 kbit/s with 4 tq per bit time) to 25 MHz (bit rate of 1 Mbit/s with 25 tq per bit time). Valid values are 4 to 25. Configured values below 4 are interpreted as 4, values above 25 are interpreted as 25.

46.3.15.3.4 Calibration Status Register (CSTAT)

Offset: 0x0008 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CALS		0	TQC											OCPC	
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OCPC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 588. Calibration Status Register (CSTAT)

Table 648. CSTAT field descriptions

Field	Description
0:1 CALS	Calibration State 00 Not_Calibrated 01 Basic_Calibrated 10 Precision_Calibrated 11 Reserved

Table 648. CSTAT field descriptions (continued)

Field	Description
3:13 TQC	Time Quanta Counter (0x000–0x3FF) Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.
14:31 OCPC	Oscillator Clock Period Counter (0x00000–0x3FFFF) Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

46.3.15.3.5 Calibration Watchdog Register (CWD)

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (CSTAT.CALS = ‘00’). In this state the calibration watchdog monitors the Message Received input. In case no message was received until the calibration watchdog has counted down to zero, the calibration FSM stays in state Not_Calibrated (CSTAT.CALS = ‘00’), the counter is reloaded with RWD.WDC and basic calibration is restarted after the next falling edge.

When in state Basic_Calibrated (CSTAT.CALS = ‘01’), the calibration watchdog is restarted with each Message Received = ‘1’). In case no message was received until the calibration watchdog has counted down to zero, the calibration FSM returns to state Not_Calibrated (CSTAT.CALS = ‘00’), the counter is reloaded with RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message was received, state Precision_Calibrated (CSTAT.CALS = ‘10’) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the Quartz Message Received input. In case no message from a quartz controlled node was received by the attached M_CAN until the calibration watchdog has counted down to zero, the calibration FSM transits back to state Basic_Calibrated (CSTAT.CALS = ‘01’).

The Calibration Watchdog Counter is clocked by the Sample Point input. The duration of a cu_spt pulse is one Time Quanta clock period. The signal is active when the CAN protocol engine on the attached M_CAN is started, that is when the INIT bit is reset.

A calibration watchdog event is signaled by Calibration Watchdog Event output. The duration of aCalibration Watchdog Event pulse is one Host clock period.

A calibration watchdog event also sets interrupt flag **CUIR.CWE**. If enabled by **CUIE.CWEE**, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag **CUIR.CWE** is reset.

Offset: 0x000C Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	WDC(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when input Configuration Change Enable input = ‘1’. Configuration Change Enable inputsignal is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Figure 589. Calibration Watchdog (CWD)

Table 649. CWD field descriptions

Field	Description
0:15 WDV	Watchdog Value Actual Calibration Watchdog Counter Value.
16:31 WDC	Watchdog Configuration Start value of the CalibrationWatchdog Counter.With the reset value of ‘00’ the counter is disabled.

46.3.15.3.6 CU Interrupt Register (CUIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. The configuration of CUIE controls whether an interrupt is generated.

Offset: 0x0010 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CSC	CWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 590. CU Interrupt Register (CUIR)

Table 650. CUIR field descriptions

Field	Description
30 CSC	Calibration State Changed 0 Calibration State unchanged 1 Calibration State has changed
31 CWE	Calibration Watchdog Event 0 No Calibration Watchdog Event 1 Calibration Watchdog Event occurred

46.3.15.3.7 CU Interrupt Enable Register (CUIE)

The settings in the CU Interrupt Enable register determine whether a status change in the CU Interrupt Register will be signalled on an interrupt line.

Offset: 0x0014 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CSCE	CWEE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 591. CU Interrupt Enable Register (CUIE)

Table 651. CUIE field descriptions

Field	Description
30 CSCE	Calibration State Changed Enable 0 Interrupt disabled 1 Interrupt enabled
31 CWEE	Calibration Watchdog Event Enable 0 Interrupt disabled 1 Interrupt enabled

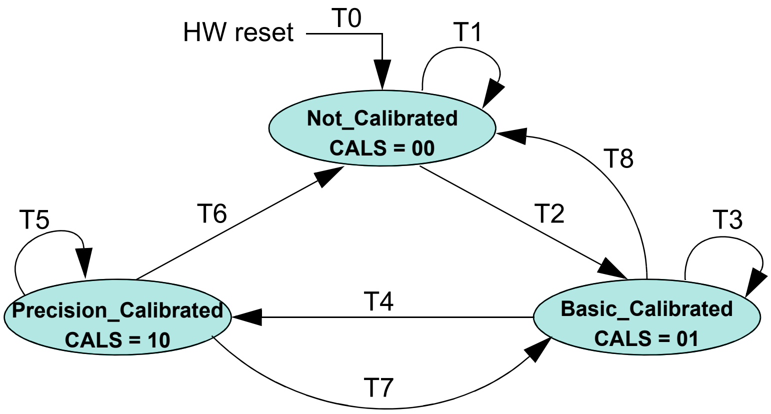
46.3.15.4 Functional Description

Calibration of the Time Quanta clock via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum tq from the clock input from Clock Select multiplexer .

- First step: Basic Calibration–The minimum distance between two edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider is updated each time a new measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol controller detects a valid CAN message. This is signaled to the Calibration on CAN unit by Message Received input signal.

- Second step: Precision Calibration – The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of clock input from Clock Select multiplexer periods. The length of this bit sequence can be configured to 32 or 64 bits via CCFG.CLF. For a calibration field length of 32/64 bit a calibration message with at least 2/6 byte data field is required. The counter is controlled by Calibration Field and Sample Point input signals. The counter is enabled when Calibration Field is set and stops after the configured number of bits has been received. Received bits are signaled by the M_CAN by Sample Point signal. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 592. Calibration FSM



T0: HW reset

T1: check for minimum

T2: Message Received = ‘1’

T3: evaluate messages

T4: Quartz Message Received = ‘1’

T5: evaluate quartz messages

T6: Configuration Change Enabled = ‘1’

T7: watchdog event: no quartz message received

T8: watchdog event: no message received
or Configuration Change Enabled = ‘1’

A change in the calibration state is signaled via by Calibration State Changed output. The duration of a Calibration State Changed output pulse is one Host clock period.

A change in the calibration state also sets interrupt flag CUIR.CSC. If enabled by CUIE.CSCE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CSC is reset.

Until precision calibration is achieved (Calibration=‘1’), the attached M_CANs operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the PLL is done by software by evaluating the calibration status from register CSTAT, the attached M_CANs have to be set to restricted operation mode (CCCR.ASM = ‘1’) until the Calibration on CAN unit is in state Precision_Calibrated (refer to *Section 46.3.15.4.3: Software Calibration*).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. Calibration frames are detected by the M_CAN’s acceptance filtering and are signaled to the Clock Calibration on CAN Unit by Quartz Message Received, to be connected to one of the M_CAN’s Filter Event outputs, signal. A filter element and an Rx Buffer have to be configured in the attached M_CAN to identify and store calibration messages. After reception of a calibration message the Rx Buffer’s new data flag has to be reset to enable signaling of the next calibration message.

In case there is only one CAN transmitter with a quartz clock in the network, this node has to transmit its first message after startup with at least one ‘1010’ sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node’s messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

Note: When the Clock Calibration on CAN Unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the attached M_CANs complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

46.3.15.4.1 Configuration

The Clock Calibration on CAN Unit is configured via register CCFG. The register is only writeable while Configuration Change Enable = ‘1’, that is when the attached M_CAN has CCCR.CCE and CCCR.INIT set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at CAN receive input pin is measured. The number of clock periods depends on the clock frequency applied at clock input from clock select multiplexer. In case the measured number of clock periods is below the minimum configured by CCFG.OCPM, for example due to a glitch on CAN receive input, the value is discarded and measurement continues.

It is recommended to configure CCFG.OCPM slightly below two CAN bit times:

CCFG.OCPM < ((2 x CAN bit time) / Clock input from Clock Select multiplexer period) / 32

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages. The reception of the bits is signaled by the attached M_CAN by Sample Point input.

The number of time quanta per bit time configured by CCFG.TQBT is used together with the measured number of oscillator clock periods CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring CCFG.BCC = ‘1’, the internal clock divider has to be configured via CCFG.CDIV to fulfill the condition CAN clock ≤ Host clock.

Note: When clock calibration on CAN is active (CCFG.BCC = ‘0’), the baud rate prescaler of the attached M_CANs has to be configured to inactive.

46.3.15.4.2 Status signaling

The status of the Clock Calibration on CAN Unit can be monitored by reading register CSTAT. When in state Precision_Calibrated the oscillator clock period counter CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by CSTAT.CALS. A change in the calibration state is signalled at the module interface by Calibration State Changed. The duration of a Calibration State Changed pulse is one Host clock period. A change in the calibration state also sets interrupt flag CUIR.CSC. If enabled by CUIE.CSCE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CSC is reset.

A calibration watchdog event is signalled by Calibration Watchdog Event output. The duration of a Calibration Watchdog Event output pulse is one Host clock period. A calibration watchdog event also sets interrupt flag CUIR.CWE. If enabled by CUIE.CWEE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CWE is reset

46.3.15.4.3 Software Calibration

The Clock Calibration on CAN Unit also supports software calibration of the clock generated from an adjustable PLL, connected to the clock input, from Clock Select multiplexer. For this purpose the clock from Clock Select multiplexer has to be routed to Time Quanta clock output by programming CCFG.BCC = ‘1’. The Clock Calibration on CAN Unit’s internal clock divider has to be configured via CCFG.CDIV to meet the clock requirements of the attached M_CANs. All other configuration parameters have to be set via CCFG as if the Clock Calibration on CAN Unit is generating Time Quanta clock.

For startup the M_CANs attached to the Clock Calibration on CAN Unit have to be configured for Restricted Operation (CCCR.ASM = ‘1’). The PLL clock has to be adjusted until the Clock Calibration on CAN Unit has reached state Precision_Calibrated. Now the software has to reset CCCR.ASM and the M_CANs can start normal operation.

During operation the software has to check regularly whether the Clock Calibration on CAN Unit is still in state Precision_Calibrated. In case the Clock Calibration on CAN Unit has left state Precision_Calibrated due to drift of the PLL clock, the attached M_CANs have to be set into Restricted Operation Mode by programming CCCR.INIT, CCCR.CCE, and CCCR.ASM to ‘1’.

After the PLL clock has been adjusted successfully (Clock Calibration on CAN Unit is in state Precision_Calibrated), the M_CANs can be set to normal operation again.

46.4 CAN RAM arbiter

This block acts as a dynamic arbiter between the various CAN nodes and the CPU for granting access to the shared CAN memory. The dynamic arbitration ensures:

- 50% bandwidth for CPU, 50% shared by the CAN nodes when simultaneous access requested from all masters and CPU,
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access.

Note: From the arbiter prospective the CPU bandwidth may be limited to 50%. The CPU may be waiting stated due to this bandwidth limitation when accessing the shared memory.

46.4.1 Features

- Active low asynchronous reset
- 1–CPU + up to 5–CAN arbiter
- Combined dynamic arbitration scheme: Bandwidth between CPU and M_CAN interfaces can be redistributed. If active, the CPU interface is granted a 50% portion of the available bandwidth
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access
- The SRAM access by a single master needs two cycles (address and data) for each read/write command. The arbiter has a pseudo address pre-fetching mechanism. This mechanism acts when multiple masters are accessing the SRAM and it overlaps the address cycle of one master with the data cycle of another master. In case of multiple masters accessing the SRAM, the pseudo address pre-fetching scheme saves multiple clock cycles
- The peripheral GSI module enable acts as a request from each master. Each master will be made to wait at least one clock cycle before the grant is given. The arbiter ensures that the CPU does not wait for more than 1 clock cycle, hence ensuring a 50% guaranteed bandwidth
- Time slot is defined as one peripheral clock cycle

46.4.2 Functional overview using examples

The dynamic arbitration scheme is explained using the following examples.

- Example 1 (active CPU IF, 3 out of 3 CAN nodes are active):
  – CPU obtains every 2nd slot
  – CAN 1/2/3 obtains every 6th time slot
- Example 2 (inactive CPU IF, 4 out of 4 CAN nodes are active):
  – CPU obtains no time slots
  – CAN 1/2/3/4 obtains every 4th time slot
- Example 3 (active CPU IF, 2 out of 4 CAN node are active):
  – CPU obtains every 2nd time slot
  – Active CAN 1/2 obtains every 4th time slot
- Example 4 (inactive CPU IF, 2 out of 4 CAN node are active):
  – CPU obtains no time slots
  – Active CAN 1/2 obtains every 2nd time slot

46.5 SRAM interface and memory organization

The CAN subsystem will interface with an external RAM using this interface. The RAM sizes required for the CAN subsystem are detailed in *Section 46.7: Shared memory map*.

46.5.1 ECC controller

The RAM embeds ECC logic and code to provide Single Error Correction / Double Error Detection (SECCDED). The module does not guarantee any proper functionality if more than 2 bits are in errors. It supports only 32-bit write access and 8-/16-/32-bit read access.

The ECC error is reported to each CAN module as well as available at the CAN subsystem top, so that the same can be forwarded to the Error Management Module.

The module uses Hamming code for single error correction and double error detection logic. It involves transmitting data with multiple ecc_check bits and decoding the associated ecc_check bits when receiving data to detect errors. The SECDED Hamming code is not able to detect three bit errors. Rather, in the presence of a three bit error, a conventional SECDED code returns an error code that is indistinguishable from an error code resulting from a single bit error. Hence, this module does not guarantee any proper functionality when more than 2 bits are in error.

The ecc_check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several ecc_check bits show parity errors after decoding the retrieved codeword. The combination of these ecc_check bit errors displays the nature of the error. In addition, the position of any single bit error is identified from the ecc_check bits.

The ECC error address is reported at the top of the subsystem using a 16 bit ECC error address signal. This address is valid only when the ECC bit error output is valid.

46.5.1.1 Features

The ECC module supports the following:

- ecc_error = ‘00’ No error is detected in the read data from memory
- ecc_error = ‘01’ This indicates single bit error occurred within the 39-bit codeword. In addition, the error is corrected, and the forwarded data to the master is error free.
- ecc_error = ‘10’ This indicates that a two bit error has occurred within the codeword. In this case, no error correction is possible.
- ecc_error = ‘11’ This indicates that errors beyond the detection capability have occurred within the codeword and no error correction is possible. This is an invalid error type.

46.6 External signal description

The CAN subsystem has the following external pins.

- M_CAN_x_RX: M_CAN x receive input
- M_CAN_x_TX: M_CAN x transmit output
- M_CAN_x_RXFD: M_CAN_x Receive Fast Data control signal
- M_CAN_x_TXFD: M_CAN_x Transmit Fast Data control signal

46.7 Shared memory map

Table 652 shows the shared memory map vs CAN subsystem transmit, receive and filters elements.

Table 652. Shared memory map versus CAN transmit, receive and filters elements

<table>
<tr><th>Start Address Offset (byte)</th><th>End Address Offset (byte)</th><th colspan="2">CAN block/sub block</th><th>Comment</th></tr>
<tr><td colspan="5">CAN Subsystem_0 - PBRIDGE_2</td></tr>
<tr><td rowspan="8">0x0000</td><td rowspan="8">0x3DFF</td><td rowspan="8">CAN_SUB_0_M_CAN_0
CAN_SUB_0_M_CAN_1
CAN_SUB_0_M_CAN_2
CAN_SUB_0_M_CAN_3</td><td>Standard Filters</td><td rowspan="8">This available physical memory space could be configured for Filters/ FIFO/Buffer.

This memory space can be flexible, assigned to any of the implemented M_CAN modules.</td></tr>
<tr><td>Extended Filters</td></tr>
<tr><td>Rx FIFO 0</td></tr>
<tr><td>Rx FIFO 1</td></tr>
<tr><td>Rx Buffer</td></tr>
<tr><td>Tx Event FIFO</td></tr>
<tr><td>Tx Buffers</td></tr>
<tr><td>Reserved</td></tr>
<tr><td colspan="5">CAN Subsystem_1 - PBRIDGE_1</td></tr>
<tr><td rowspan="8">0x0000</td><td rowspan="8">0x3DFF</td><td rowspan="8">CAN_SUB_1_M_CAN_1
CAN_SUB_1_M_CAN_2
CAN_SUB_1_M_CAN_3
CAN_SUB_1_M_CAN_4</td><td>Standard Filters</td><td rowspan="8">This available physical memory space could be configured for Filters/ FIFO/Buffer.

This memory space can be flexible, assigned to any of the implemented M_CAN modules.</td></tr>
<tr><td>Extended Filters</td></tr>
<tr><td>Rx FIFO 0</td></tr>
<tr><td>Rx FIFO 2</td></tr>
<tr><td>Rx Buffer</td></tr>
<tr><td>Tx Event FIFO</td></tr>
<tr><td>Tx Buffers</td></tr>
<tr><td>Reserved</td></tr>
</table>

Note: There is no write access protection from a CAN module to the message memory space.

The start address for every memory section is to be configured by the user in the relevant CAN modules registers. However there is no hardware check in the RAM interface for the start address consistency.

However the number of filters and the Tx/Rx buffers used, it is mandatory for the user to configure the start address registers, specified in Table 653, in order to fit the memory mapping specified in Table 652.

The CPU uses the peripheral interface to access the memory locations. It must be noted that there is no transfer error reported to the CPU in case of access to reserved memory locations.

Table 653. Detailed mapping in CAN sub-blocks

CAN sub-block	Element name	Comment
Standard Filters	Standard Message ID filter 0 ... Standard Message ID filter 127	address configured in SIDFC [FLSSA]
Extended Filters	Extended Message ID filter 0 ... Extended Message ID filter 63	address configured in XIDFC [FLESA]
Rx FIFO0	Rx FIFO0 element 0 ... Rx FIFO0 element 63	address configured in RXF0C[F0SA]
Rx FIFO1	Rx FIFO1 element 0 ... Rx FIFO1 element 63	address configured in RXF1C[F1SA]
Rx Buffer	Rx Buffer element 0 ... Rx Buffer element 63	address configured in RXBC[RBSA]
Tx Event FIFO	Tx event FIFO element 0 ... Tx event FIFO element 31	address configured in TXEFC[EFSA]
Tx Buffers	Tx Buffer element 0 ... Tx Buffer element 31	address configured in TXBC[TBSA]

47 Ethernet

47.1 Overview

The Ethernet module enables a host to transmit and receive data over the Ethernet in compliance with the IEEE 802.3-2008. *Figure 593* shows a system-level block diagram of the Ethernet module.

As shown in *Figure 593*, the Ethernet module uses a 64-bit AMBA High-Performance Bus (AHB) on the application side. The AHB interface transfers the data to and from system memory through AHB master interface. One AHB master interface is connected to all DMA channels. The DMA arbiter helps in arbitration of all the paths (Transmit and Receive) in all channels.

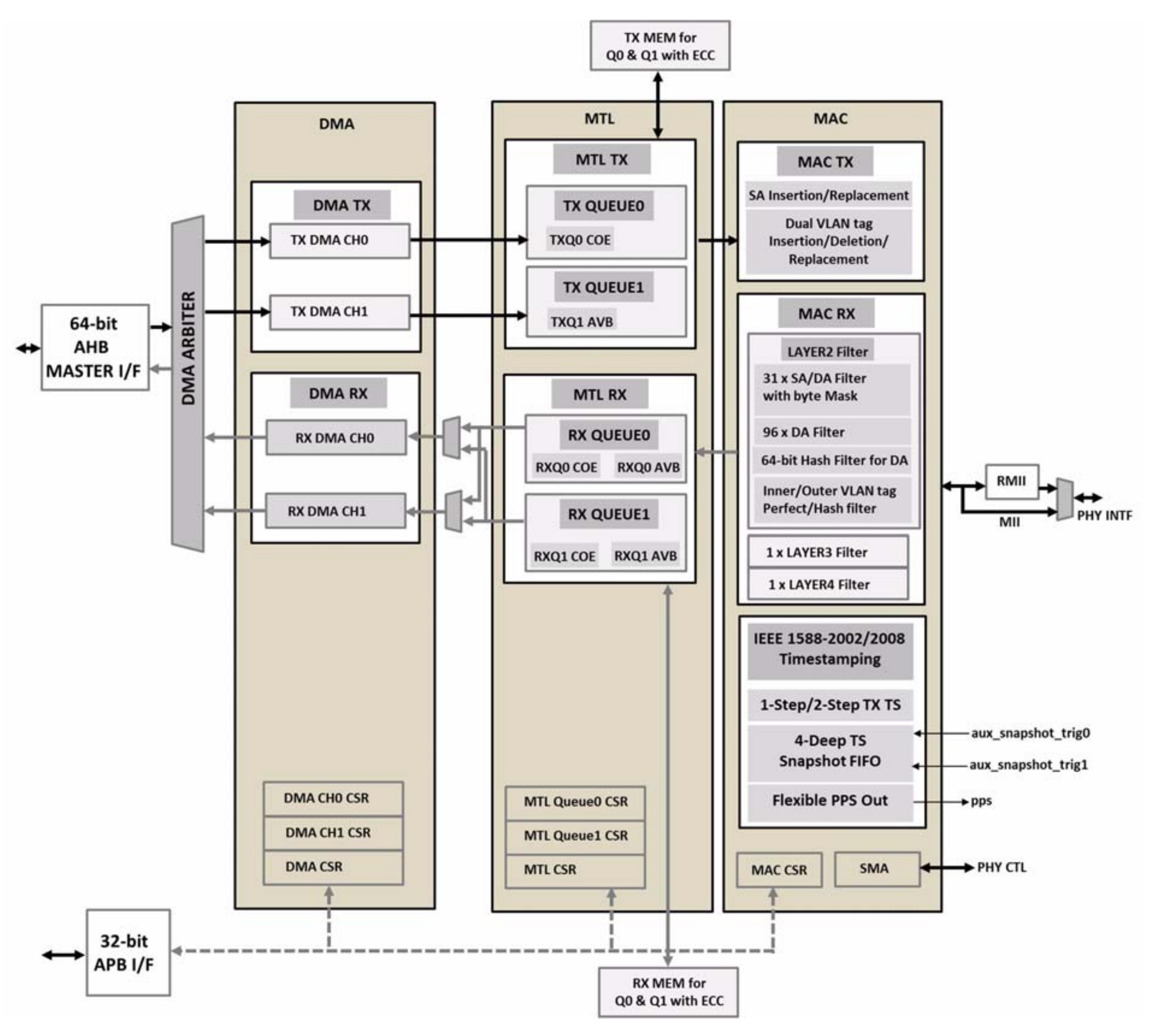
As shown in *Figure 593*, the Ethernet module uses a 32-bit APB interface to program the Control and Status registers (CSR). Each channel has a separate set of Control and Status registers (CSR) for managing the Transmit and Receive functions, descriptor handling, and interrupt handling.

The Transmit FIFO (Tx FIFO) buffers the data transferred from the application to the Ethernet module. Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until they can be transferred to the application. These are asynchronous FIFOs because they also transfer the data between the application clock and the MAC line clocks. Tx memory and Rx memory are one-ported RAM of 68-bit width for 64 data bus widths. The extra bits are used for storing the byte-enable information.

For multiple queues, all Tx queue share the Tx FIFO memory and all Rx queue share the Rx FIFO memory. The application can program the size of the FIFO memory allocated to each Tx or Rx queue. The Ethernet module provides a direct read or write access to the FIFO memories in Debug mode.

The Ethernet module can be programed to use either of MII or RMII PHY interfaces.

Figure 593. System Level block diagram



47.1.1 Ethernet module Features

47.1.1.1 Standard Compliance

The Ethernet module is compliant with the following standards:

- IEEE 802.3-2008 for Ethernet MAC, Media Independent Interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Audio Video (AV) traffic
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB master and APB slave ports
- RMII specification version 1.2 from RMII consortium

47.1.1.2 MAC

The MAC supports the following features:

- MAC Tx and Rx features
- MAC Tx features
- MAC Rx features

47.1.1.2.1 MAC Tx and Rx features

Following is the list of MAC Tx and Rx features:

- Separate transmission, reception, and control interfaces to the application
- 10 and 100, Mbps data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
  - RMII interface to communicate with an external Fast Ethernet PHY
- Half-duplex operation:
  - CSMA/CD Protocol support
  - Flow control using backpressure support
  - Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII PHYs.
- 64-bit data transfer interface on the application side
- Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
- Network statistics with RMON or MIB Counters (RFC2819/RFC2665)
- Support for Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping is supported in TX direction
- Flexibility to control the Pulse-Per-Second (PPS) output signal
- MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

47.1.1.2.2 MAC Tx features

Following is the list of MAC Tx features:

- Preamble and start of packet data (SFD) insertion
- Separate 32-bit status for each packet transmitted from the application
- Automatic CRC and pad generation controllable on a per-packet basis
- Programmable packet length to support Standard or Jumbo Ethernet packets with up to 16 KB of size
- Programmable Inter Packet Gap (40–96 bit times in steps of 8)
- IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in full-duplex mode)
- Source Address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in full-duplex mode

47.1.1.2.3 MAC Rx features

Following is the list of MAC Rx features:

- Automatic Pad and CRC Stripping options
- Option to disable Automatic CRC checking
- Preamble and SFD deletion
- Separate 112-bit or 128-bit status
- Programmable watchdog timeout limit
- Flexible address filtering modes:
  - 128 MAC addresses for unicast 48-bit perfect filtering
    - 31 filters for destination address (DA) or source address (SA) comparison with masks for each byte
    - 97 filters for destination address (DA) that can be selected in blocks of 32 and 64 registers
  - 64 bit Hash filter for multicast and unicast (DA) addresses
  - Option to pass all multicast addressed packets
  - Promiscuous mode to pass all packets without any filtering for network monitoring
  - Pass all incoming packets (as per filter) with a status report
- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based filtering. Filtering based on either outer or inner VLAN tag is possible
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- module to detect remote wake-up packets and AMD magic packets
- forwarding of received Pause packets to the application (in full-duplex mode)
- Receive module for Layer 3/Layer 4 checksum offload for received packets
- Stripping of up to two VLAN Tags and providing the tags in the status

47.1.1.3 MAC Transaction Layer (MTL)

The MTL block consists of the following FIFOs: Tx FIFO and Rx FIFO. The FIFO space is shared by multiple queues. The buffer size is configured for each queue in multiples of 256 bytes.

47.1.1.3.1 MTL Tx and Rx features

Following is the list of MTL Tx and Rx features:

- 64-bit Transaction Layer block (bridges the application and the MAC)
- Data transfers executed using simple FIFO protocol
- Synchronization for all clocks in the design (Transmit, Receive, and Application clocks)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable threshold capability for each queue (default of 64 bytes)
- Debug and slave mode operation on Queue 0 (default queue)

47.1.1.3.2 MTL Tx features

Following is the list of MTL Tx features:

- Two single-port RAM-based (SPRAM) synchronous FIFO controllers
- 4 KB TX FIFO size
- Dual queues on the Transmit path with a common memory for both Tx queues
- Store-and-Forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Programmable queue size. Each queue size can be programed in terms of 256 bytes
- Automatic retransmission of collision packets in half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Disabling of Data Memory RAM chip-select when inactive to reduce power consumption
- Checksum offload engine calculates and inserts IPv4 header checksum and TCP, UDP, or ICMP checksum
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Programmable interrupt options for different operational conditions
- Option to support dropping of Tx Status to improve the Transmit throughput
- Packet-level control for:
  - VLAN tag insertion or replacement
  - Ethernet source address insertion
  - Layer3/Layer4 Checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control
- Following scheduling algorithms in configurations with multiple queues:
  - Weighted Round Robin (WRR)
  - Strict Priority (SP)
  - Credit-based Shaper (CBS) for AVB
- Statistics related to bandwidth consumption by each queue of up to 16 blocks over a 125 µs period

47.1.1.3.3 MTL Rx features

Following is the list of MTL Rx features:

- 8 KB Rx FIFO size
- Dual queues on the Receive path with a common memory for all Rx queues
- Programmable Rx queue threshold (default fixed at 64 bytes) in Threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of Pause packet control or backpressure signal to the MAC based on the Rx Queue fill level
- Arbitration among queues when multiple queues are present. The following arbitration schemes are supported:
  - Weighted Round Robin (WRR)
  - Weighted Strict priority (WSP)
  - Strict Priority (SP)

47.1.1.4 DMA block

The DMA block exchanges data between the MTL block and system memory. The well-defined descriptors structure acts as a software and hardware interface. The application can use a set of registers (DMA CSR) to control the DMA operations. The DMA block supports the following features:

- 64-bit data transfers
- 2-channel Transmit and 2-channel Receive engines
- Separate DMA channel in the Transmit path for each queue in MTL
- 2-DMA channels for queues in MTL Receive path
- Fully synchronous design operating on a single application clock except for CSR module
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture to allow large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 KB of data)
- Comprehensive status reporting for normal operation and transfers with errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions

- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Start and Stop modes
- Separate ports for host CSR access and host data interface
- Routing of received packets to the RX DMA channels based on the DA or VLAN Priority in multi-channel DMA configurations
- Time-sensitive conditional packet transmission by comparing the Slot Time information provided in the descriptor (useful for AV applications)

47.1.1.5 AMBA AHB Master Interface

The AHB master interface supports the following features:

- Interfaces with the application through 64-bit AHB
- Option to select address-aligned bursts from AHB master port
- Split, Retry, and Error AHB responses
- AHB 1 K boundary burst splitting
- Software-selected type of AHB burst (fixed burst, indefinite burst, or mix of both)

The AHB master interface does not generate the following:

- Wrap burst
- Locked or Protected transfers

47.1.1.6 AMBA APB Slave Interface

The Ethernet module supports the 32-bit APB slave interface without any byte-enable to program the CSR registers.

47.1.1.7 Audio Video Support

The Ethernet module supports the following Audio Video (AV) features:

- Separate channels or queues for AV data transfer in 100 Mbps modes
- Up to 2-queues on the Receive paths for AV traffic and 1-queue on the Transmit path for AV traffic
- IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for Transmit channels
- Single Tx FIFO and Rx FIFO (MTL) for all selected queues (system-side interface [AHB] remains the same)

47.1.1.8 Monitoring, Testing, and Debugging Support

The Ethernet module supports the following features for monitoring, testing, and debugging:

- Internal loopback from Tx to Rx on the MII for debugging
- DMA states (Tx and Rx) as status bits
- Debug status register that gives status of FSMs in Transmit and Receive data paths and FIFO fill-levels
- Application Abort status bits
- MMC (RMON) module
- Current Tx or Rx Buffer pointer as status registers
- Current Tx or Rx Descriptor pointer as status registers
- Statistical counters to calculate the bandwidth served by each Transmit channel when AV support is enabled
- Tx or Rx Queues memory accessible through Slave port for debug

47.2 Memory map and register description

This section provides a detailed description of all registers accessible in this module.

Table 654. Ethernet memory map

Address offset (hex)	Register	Location
0x0000	MAC Configuration register (MAC_CONFIGURATION)	*Section 47.2.1*
0x0004	MAC Extended Configuration register (MAC_EXT_CONFIGURATION)	*Section 47.2.2*
0x0008	MAC Packet Filter register (MAC_PACKET_FILTER)	*Section 47.2.3*
0x000C	Watchdog Timeout register (MAC_WATCHDOG_TIMEOUT)	*Section 47.2.4*
0x0010	Hash Table register 0 (MAC_HASH_TABLE_REG0)	*Section 47.2.5*
0x0014	Hash Table register 1 (MAC_HASH_TABLE_REG1)	*Section 47.2.6*
0x0018–0x004F	Reserved	
0x0050	VLAN Tag register (MAC_VLAN_TAG)	*Section 47.2.7*
0x0054–0x0057	Reserved	
0x0058	VLAN Hash Table register (MAC_VLAN_HASH_TABLE)	*Section 47.2.8*
0x005C–0x005F	Reserved	
0x0060	VLAN Tag Inclusion register (MAC_VLAN_INCL)	*Section 47.2.9*
0x0064	Inner VLAN Tag Inclusion register (MAC_INNER_VLAN_INCL)	*Section 47.2.10*
0x0068–0x006F	Reserved	
0x0070	MAC Q0 Flow Control register (MAC_Q0_TX_FLOW_CTRL)	*Section 47.2.11*
0x0074	MAC Q1 Flow Control register (MAC_Q1_TX_FLOW_CTRL)	*Section 47.2.12*
0x0078–0x008F	Reserved	
0x0090	MAC Receive Flow Control register (MAC_RX_FLOW_CTRL)	*Section 47.2.13*
0x0094–0x0097	Reserved	

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0098	Transmit Queue Priority Mapping 0 register (MAC_TXQ_PRTY_MAP0)	Section 47.2.14
0x009C–0x009F	Reserved	
0x00A0	Receive Queue Control 0 register (MAC_RXQ_CTRL0)	Section 47.2.15
0x00A4	Receive Queue Control 1 register (MAC_RXQ_CTRL1)	Section 47.2.16
0x00A8	Receive Queue Control 2 register (MAC_RXQ_CTRL2)	Section 47.2.17
0x00AC–0x00AF	Reserved	
0x00B0	Interrupt Status register (MAC_INTERRUPT_STATUS)	Section 47.2.18
0x00B4	Interrupt Enable register (MAC_INTERRUPT_ENABLE)	Section 47.2.19
0x00B8	Receive Transmit Status register (MAC_RX_TX_STATUS)	Section 47.2.20
0x00BC–0x00BF	Reserved	
0x00C0	PMT Control and Status register (MAC_PMT_CONTROL_STATUS)	Section 47.2.21
0x00C4	Wake-up Packet Filter registers	Section 47.2.22
0x00C8–0x00CF	Reserved	
0x00D0	LPI Control and Status register (MAC_LPI_CONTROL_STATUS)	Section 47.2.23
0x00D4	LPI Timers Control register (MAC_LPI_TIMERS_CONTROL)	Section 47.2.24
0x00D8	LPI Entry Timer register (MAC_LPI_ENTRY_TIMER)	Section 47.2.25
0x00DC	1US Tic Counter register (MAC_1US_TIC_COUNTER)	Section 47.2.26
0x00E0–0x010F	Reserved	
0x0110	Version register (MAC_VERSION)	Section 47.2.27
0x0114	Debug register (MAC_DEBUG)	Section 47.2.28
0x0118–0x011B	Reserved	
0x011C	Hardware Feature0 register (MAC_HW_FEATURE0)	Section 47.2.29
0x0120	Hardware Feature1 register (MAC_HW_FEATURE1)	Section 47.2.30
0x0124	Hardware Feature2 register (MAC_HW_FEATURE2)	Section 47.2.31
0x0128–0x01FF	Reserved	
0x0200	MDIO Address register (MAC_MDIO_ADDRESS)	Section 47.2.32
0x0204	MDIO Data register (MAC_MDIO_DATA)	Section 47.2.33
0x0208–0x021F	Reserved	
0x0210	ARP Address register (MAC_ARP_ADDRESS)	Section 47.2.34
0x0214–0x02FF	Reserved	
0x0300	Address0 High register (MAC_ADDRESS0_HIGH)	Section 47.2.35
0x0304	Address0 Low register (MAC_ADDRESS0_LOW)	Section 47.2.36
(for *n* = 1 to 31)		
0x0300 + *n**0x8	MAC Address High register 1 to 31 **(**MAC_ADDRESS*n*_HIGH31_1)	Section 47.2.37

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0304 + *n**0x8	MAC Address Low register 1 to 31 (MAC_ADDRESS*n*_LOW31_1)	Section 47.2.38
(for *n* = 32 to 63)		
0x0300 + *n**0x8	MAC Address High register 32 to 63 (MAC_ADDRESS*n*_HIGH63_32)	Section 47.2.39
0x0304 + *n**0x8	MAC Address Low register 32 to 63 (MAC_ADDRESS*n*_LOW63_32)	Section 47.2.40
(for *n* = 64 to 127)		
0x0300 + n*0x8	MAC Address High register 64 to 127 (MAC_ADDRESS*n*_HIGH127_64)	Section 47.2.41
0x0304 + *n**0x8	MAC Address Low register 64 to 127 (MAC_ADDRESS*n*_LOW127_64)	Section 47.2.42
0x0700	MMC Control register (MMC_CONTROL)	Section 47.2.43
0x0704	MMC Receive Interrupt register (MMC_RX_INTERRUPT)	Section 47.2.44
0x0708	MMC Tx Interrupt register (MMC_TX_INTERRUPT)	Section 47.2.45
0x070C	MMC Rx Interrupt mask register (MMC_RX_INTERRUPT_MASK)	Section 47.2.46
0x0710	MMC Tx Interrupt Mask register (MMC_TX_INTERRUPT_MASK)	Section 47.2.47
0x0714	Transmit Octet Count Good Bad register (TX_OCTET_COUNT_GOOD_BAD)	Section 47.2.48
0x0718	Transmit Packet Count Good Bad register (TX_PACKET_COUNT_GOOD_BAD)	Section 47.2.49
0x071C	Transmit Broadcast Packets Good register (TX_BROADCAST_PACKETS_GOOD)	Section 47.2.50
0x0720	Transmit Multicast Packets Good register (TX_MULTICAST_PACKETS_GOOD)	Section 47.2.51
0x0724	Transmit 64Octets Packets Good Bad register (TX_64OCTETS_PACKETS_GOOD_BAD)	Section 47.2.52
0x0728	Transmit 65to127Octets Packets Good Bad register (TX_65TO127OCTETS_PACKETS_GOOD_BAD)	Section 47.2.53
0x072C	Transmit 128To255Octets Packets Good Bad register (TX_128TO255OCTETS_PACKETS_GOOD_BAD)	Section 47.2.54
0x0730	Transmit 256To511Octets Packets Good Bad register (TX_256TO511OCTETS_PACKETS_GOOD_BAD)	Section 47.2.55
0x0734	Transmit 512To1023Octets Packets Good Bad register (TX_512TO1023OCTETS_PACKETS_GOOD_BAD)	Section 47.2.56
0x0738	Transmit 1024ToMaxOctets Packets Good Bad register (TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)	Section 47.2.57
0x073C	Transmit Unicast Packets Good Bad register (TX_UNICAST_PACKETS_GOOD_BAD)	Section 47.2.58
0x0740	Transmit Multicast Packets Good Bad register (TX_MULTICAST_PACKETS_GOOD_BAD)	Section 47.2.59
0x0744	Transmit Broadcast Packets Good Bad register (TX_BROADCAST_PACKETS_GOOD_BAD)	Section 47.2.60

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0748	Transmit Underflow Error Packets register (TX_UNDERFLOW_ERROR_PACKETS)	Section 47.2.61
0x074C	Transmit Single Collision Good Packets register (TX_SINGLE_COLLISION_GOOD_PACKETS)	Section 47.2.62
0x0750	Transmit Multiple Collision Good Packets register (TX_MULTIPLE_COLLISION_GOOD_PACKETS)	Section 47.2.63
0x0754	Transmit Deferred Packets register (TX_DEFERRED_PACKETS)	Section 47.2.64
0x0758	Transmit Late Collision Packets register (TX_LATE_COLLISION_PACKETS)	Section 47.2.65
0x075C	Transmit Excessive Collision Packets register (TX_EXCESSIVE_COLLISION_PACKETS)	Section 47.2.66
0x0760	Transmit Carrier Error Packets register (TX_CARRIER_ERROR_PACKETS)	Section 47.2.67
0x0764	Transmit Octet Count Good register (TX_OCTET_COUNT_GOOD)	Section 47.2.68
0x0768	Transmit Packet Count Good register (TX_PACKET_COUNT_GOOD)	Section 47.2.69
0x076C	Transmit Excessive Deferral Error register (TX_EXCESSIVE_DEFERRAL_ERROR)	Section 47.2.70
0x0770	Transmit Pause Packets register (TX_PAUSE_PACKETS)	Section 47.2.71
0x0774	Transmit VLAN Packets Good register (TX_VLAN_PACKETS_GOOD)	Section 47.2.72
0x0778	Transmit OSize Packets Good register (TX_OSIZE_PACKETS_GOOD)	Section 47.2.73
0x077C–0x077F	Reserved	
0x0780	Receive Packets Count Good Bad register (RX_PACKETS_COUNT_GOOD_BAD)	Section 47.2.74
0x0784	Receive Octet Count Good Bad register (RX_OCTET_COUNT_GOOD_BAD)	Section 47.2.75
0x0788	Receive Octet Count Good register (RX_OCTET_COUNT_GOOD)	Section 47.2.76
0x078C	Receive Broadcast Packets Good register (RX_BROADCAST_PACKETS_GOOD)	Section 47.2.77
0x0790	Receive Multicast Packets Good register (RX_MULTICAST_PACKETS_GOOD)	Section 47.2.78
0x0794	Receive CRC Error Packets register (RX_CRC_ERROR_PACKETS)	Section 47.2.79
0x0798	Receive Alignment Error Packets register (RX_ALIGNMENT_ERROR_PACKETS)	Section 47.2.80
0x079C	Receive Runt Error Packets register (RX_RUNT_ERROR_PACKETS)	Section 47.2.81
0x07A0	Receive Jabber Error Packets register (RX_JABBER_ERROR_PACKETS)	Section 47.2.82
0x07A4	Receive Undersize Packets Good register (RX_UNDERSIZE_PACKETS_GOOD)	Section 47.2.83

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x07A8	Receive Oversize Packets Good register (RX_OVERSIZE_PACKETS_GOOD)	Section 47.2.84
0x07AC	Receive 64Octets Packets Good Bad register (RX_64OCTETS_PACKETS_GOOD_BAD)	Section 47.2.85
0x07B0	Receive 65To127Octets Packets Good Bad register (RX_65TO127OCTETS_PACKETS_GOOD_BAD)	Section 47.2.86
0x07B4	Receive 128To255Octets Packets Good Bad register (RX_128TO255OCTETS_PACKETS_GOOD_BAD)	Section 47.2.87
0x07B8	Receive 256To511Octets Packets Good Bad register (RX_256TO511OCTETS_PACKETS_GOOD_BAD)	Section 47.2.88
0x07BC	Receive 512To1023Octets Packets Good Bad register (RX_512TO1023OCTETS_PACKETS_GOOD_BAD)	Section 47.2.89
0x07C0	Receive 1024ToMax Octets Packets Good bad register (RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)	Section 47.2.90
0x07C4	Receive Unicast Packets Good register (RX_UNICAST_PACKETS_GOOD)	Section 47.2.91
0x07C8	Receive Length Error Packets register (RX_LENGTH_ERROR_PACKETS)	Section 47.2.92
0x07CC	Receive Out Of Range Type Packets register (RX_OUT_OF_RANGE_TYPE_PACKETS)	Section 47.2.93
0x07D0	Receive Pause Packets register (RX_PAUSE_PACKETS)	Section 47.2.94
0x07D4	Receive FIFO Overflow Packets register (RX_FIFO_OVERFLOW_PACKETS)	Section 47.2.95
0x07D8	Receive VLAN Packets Good Bad register (RX_VLAN_PACKETS_GOOD_BAD)	Section 47.2.96
0x07DC	Receive Watchdog Error Packets register (RX_WATCHDOG_ERROR_PACKETS)	Section 47.2.97
0x07E0	Receive Error Packets register (RX_RECEIVE_ERROR_PACKETS)	Section 47.2.98
0x07E4	Receive Control Packets Good register (RX_CONTROL_PACKETS_GOOD)	Section 47.2.99
0x07E8–0x07EB	Reserved	
0x07EC	Transmit LPI USEC Counter register (TX_LPI_USEC_CNTR)	Section 47.2.100
0x07F0	Transmit LPI Transaction Counter register (TX_LPI_TRAN_CNTR)	Section 47.2.101
0x07F4	Receive LPI USEC Counter register (RX_LPI_USEC_CNTR)	Section 47.2.102
0x07F8	Receive LPI Transaction Counter register (RX_LPI_TRAN_CNTR)	Section 47.2.103
0x07FC–0x07FF	Reserved	
0x0800	MMC IPC Receive Interrupt Mask register (MMC_IPC_RX_INTERRUPT_MASK)	Section 47.2.104
0x0804–0x0807	Reserved	

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0808	MMC IPC Receive Interrupt register (MMC_IPC_RX_INTERRUPT)	Section 47.2.105
0x080C–0x080F	Reserved	
0x0810	Receive IPv4 Good Packets register (RXIPV4_GOOD_PACKETS)	Section 47.2.106
0x0814	Receive IPv4 Header Error Packets (RXIPV4_HEADER_ERROR_PACKETS)	Section 47.2.107
0x0818	Receive IPv4 No Payload Packets register (RXIPV4_NO_PAYLOAD_PACKETS)	Section 47.2.108
0x081C	Receive IPv4 Fragmented Packets register (RXIPV4_FRAGMENTED_PACKETS)	Section 47.2.109
0x0820	Receive IPv4 UDP Checksum Disabled Packets register (RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS)	Section 47.2.110
0x0824	Receive IPv6 Good Packets register (RXIPV6_GOOD_PACKETS)	Section 47.2.111
0x0828	Receive IPv6 Header Error Packets register (RXIPV6_HEADER_ERROR_PACKETS)	Section 47.2.112
0x082C	Receive IPv6 Payload Packets register (RXIPV6_NO_PAYLOAD_PACKETS)	Section 47.2.113
0x0830	Receive UDP Good Packets register (RXUDP_GOOD_PACKETS)	Section 47.2.114
0x0834	Receive UDP Error Packets register (RXUDP_ERROR_PACKETS)	Section 47.2.115
0x0838	Receive TCP Good Packets register (RXTCP_GOOD_PACKETS)	Section 47.2.116
0x083C	Receive TCP Error Packets register (RXTCP_ERROR_PACKETS)	Section 47.2.117
0x0840	Receive ICMP Good Packets register (RXICMP_GOOD_PACKETS)	Section 47.2.118
0x0844	Receive ICMP Error Packets register (RXICMP_ERROR_PACKETS)	Section 47.2.119
0x0848–0x084F	Reserved	
0x0850	Receive IPv4 Good Octets register (RXIPV4_GOOD_OCTETS)	Section 47.2.120
0x0854	Receive IPv4 Header Error Octets register (RXIPV4_HEADER_ERROR_OCTETS)	Section 47.2.121
0x0858	Receive IPv4 No Payload Octets register (RXIPV4_NO_PAYLOAD_OCTETS)	Section 47.2.122
0x085C	Receive IPv4 Fragmented Octets register (RXIPV4_FRAGMENTED_OCTETS)	Section 47.2.123
0x0860	Receive IPv4 UDP Checksum Disable Octets register (RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS)	Section 47.2.124
0x0864	Receive IPv6 Good Octets register (RXIPV6_GOOD_OCTETS)	Section 47.2.125
0x0868	Receive Header Error Octets register (RXIPV6_HEADER_ERROR_OCTETS)	Section 47.2.126
0x086C	Receive IPv6 No Payload Octets register (RXIPV6_NO_PAYLOAD_OCTETS)	Section 47.2.127
0x0870	Receive UDP Good Octets register (RXUDP_GOOD_OCTETS)	Section 47.2.128

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0874	Receive UDP Error Octets register (RXUDP_ERROR_OCTETS)	Section 47.2.129
0x0878	Receive TCP Good Octets register (RXTCP_GOOD_OCTETS)	Section 47.2.130
0x087C	Receive TCP Error Octets register (RXTCP_ERROR_OCTETS)	Section 47.2.131
0x0880	Receive ICMP Good Octets register (RXICMP_GOOD_OCTETS)	Section 47.2.132
0x0884	Receive ICMP Error Octets register (RXICMP_ERROR_OCTETS)	Section 47.2.133
0x0888–0x088F	Reserved	
0x0900	Layer 3 and Layer 4 Control 0 register (MAC_L3_L4_CONTROL0)	Section 47.2.134
0x0904	Layer 4 Address 0 register (MAC_LAYER4_ADDRESS0)	Section 47.2.135
0x0908–0x090F	Reserved	
0x0910	Layer 3 Address 0 Register 0 (MAC_LAYER3_ADDRESS0_REG0)	Section 47.2.136
0x0914	Layer3 Address1 Register 0 (MAC_LAYER3_ADDRESS1_REG0)	Section 47.2.137
0x0918	Layer3 Address2 Register 0 (MAC_LAYER3_ADDRESS2_REG0)	Section 47.2.138
0x091C	Layer3 Address3 Register 0 (MAC_LAYER3_ADDRESS3_REG0)	Section 47.2.139
0x0920–0x0AFF	Reserved	
0x0B00	Timestamp Control register (MAC_TIMESTAMP_CONTROL)	Section 47.2.140
0x0B04	Sub Second Increment register (MAC_SUB_SECOND_INCREMENT)	Section 47.2.141
0x0B08	System Time Seconds register (MAC_SYSTEM_TIME_SECONDS)	Section 47.2.142
0x0B0C	System Time Nanoseconds register (MAC_SYSTEM_TIME_NANOSECONDS)	Section 47.2.143
0x0B10	System Time Seconds Update register (MAC_SYSTEM_TIME_SECONDS_UPDATE)	Section 47.2.144
0x0B14	System Time Nanoseconds Update register (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE)	Section 47.2.145
0x0B18	Timestamp Addend register (MAC_TIMESTAMP_ADDEND)	Section 47.2.146
0x0B1C	System Time Higher Word Seconds register (MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS)	Section 47.2.147
0x0B20	MAC Timestamp Status register (MAC_TIMESTAMP_STATUS)	Section 47.2.148
0x0B24–0x0B2F	Reserved	
0x0B30	MAC Transmit Timestamp Status Nanoseconds register (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS)	Section 47.2.149
0x0B34	MAC Tx Transmit Timestamp Status Seconds register (MAC_TX_TIMESTAMP_STATUS_SECONDS)	Section 47.2.150
0x0B38–0x0B3F	Reserved	
0x0B40	MAC Auxiliary Control register (MAC_AUXILIARY_CONTROL)	Section 47.2.151
0x0B44–0x0B47	Reserved	
0x0B48	MAC Auxiliary Timestamp Nanoseconds register (MAC_AUXILIARY_TIMESTAMP_NANOSECONDS)	Section 47.2.152

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0B4C	MAC Auxiliary Timestamp Seconds register (MAC_AUXILIARY_TIMESTAMP_SECONDS)	Section 47.2.153
0x0B50	MAC Timestamp Ingress Asymmetry Correction register (MAC_TIMESTAMP_INGRESS_ASYM_CORR)	Section 47.2.154
0x0B54	MAC Timestamp Egress Asymmetry Correction register (MAC_TIMESTAMP_EGRESS_ASYM_CORR)	Section 47.2.155
0x0B58	MAC Timestamp Ingress Correction Nanosecond register (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND)	Section 47.2.156
0x0B5C	MAC Timestamp Egress Correction Nanosecond register (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND)	Section 47.2.157
0x0B60–0x0B6F	Reserved	
0x0B70	MAC_PPS_Control PPS Control register (MAC_PPS_CONTROL)	Section 47.2.158
0x0B74–0x0B7F	Reserved	
0x0B80	PPS0 Target Time Seconds register (MAC_PPS0_TARGET_TIME_SECONDS)	Section 47.2.159
0x0B84	PPS0 Target Time Nanoseconds register (MAC_PPS0_TARGET_TIME_NANOSECONDS)	Section 47.2.160
0x0B88	PPS0 Interval register (MAC_PPS0_INTERVAL)	Section 47.2.161
0x0B8C	PPS0 Width register (MAC_PPS0_WIDTH)	Section 47.2.162
0x0B90–0x0BFF	Reserved	
MTL registers		
0x0C00	Operation Mode register (MTL_OPERATION_MODE)	Section 47.2.163
0x0C04–0x0C07	Reserved	
0x0C08	Debug Access Control register (MTL_DBG_CTL)	Section 47.2.164
0x0C0C	Debug Status register (MTL_DBG_STS)	Section 47.2.165
0x0C10	FIFO Debug Data register (MTL_FIFO_DEBUG_DATA)	Section 47.2.166
0x0C14–0x0C1F	Reserved	
0x0C20	Interrupt Status register (MTL_INTERRUPT_STATUS)	Section 47.2.167
0x0C24–0xC2F	Reserved	
0x0C30	Receive Queue and DMA Channel Mapping 0 register (MTL_RXQ_DMA_MAP0)	Section 47.2.168
0x0C34–0x0CFF	Reserved	
MTL_Q0 registers		
0x0D00	Queue 0 Transmit Operation Mode register (MTL_TXQ0_OPERATION_MODE)	Section 47.2.169
0x0D04	Queue 0 Underflow Counter register (MTL_TXQ0_UNDERFLOW)	Section 47.2.170
0x0D08	Queue 0 Transmit Debug register (MTL_TXQ0_DEBUG)	Section 47.2.171

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0D0C–0x0D13	Reserved	
0x0D14	Queue 0 ETS Status register (MTL_TXQ0_ETS_STATUS)	Section 47.2.172
0x0D18	Queue 0 Quantum or Weights register (MTL_TXQ0_QUANTUM_WEIGHT)	Section 47.2.173
0x0D1C–0x0D2B	Reserved	
0x0D2C	Interrupt Control Status register (MTL_Q0_IINTERRUPT_CONTROL_STATUS)	Section 47.2.174
0x0D30	Queue 0 Receive Operation Mode register (MTL_RXQ0_OPERATION_MODE)	Section 47.2.175
0x0D34	Queue 0 Missed Packet and Overflow Counter register (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT)	Section 47.2.176
0x0D38	Queue 0 Receive Debug register (MTL_RXQ0_DEBUG)	Section 47.2.177
0x0D3C	Queue Receive Control register (MTL_RXQ0_CONTROL)	Section 47.2.178
MTL_Q*n* registers (for *n* = 1)		
0x0D00 + *n**0x40	Transmit Q*n* Operation Mode register (MTL_TXQ*n*_OPERATION_MODE)	Section 47.2.179
0x0D04 + *n**0x40	Transmit Q*n* Underflow register (MTL_TXQ*n*_UNDERFLOW)	Section 47.2.180
0x0D08 + *n**0x40	Transmit Q*n* Debug register (MTL_TXQ*n*_DEBUG)	Section 47.2.181
0x0D0C + *n**0x40 –0x0D0F + *n**0x40	Reserved	
0x0D10 + *n**0x40	Transmit Q*n* ETS Control register (MTL_TXQ*n*_ETS_CONTROL)	Section 47.2.182
0x0D14 + *n**0x40	Transmit Q*n* ETS Status register (MTL_TXQ*n*_ETS_STATUS)	Section 47.2.183
0x0D18 + *n**0x40	Transmit Q*n* Quantum Weight register (MTL_TXQ*n*_QUANTUM_WEIGHT)	Section 47.2.184
0x0D1C + *n**0x40	Transmit Q*n* Send Slope Credit register (MTL_TXQ*n*_SENDSPLOPECREDIT)	Section 47.2.185
0x0D20 + *n**0x40	Transmit Q*n* High Credit register (MTL_TXQ*n*_HICREDIT)	Section 47.2.186
0x0D24 + *n**0x40	Transmit Q*n* Low Credit register (MTL_TXQ*n*_LOCREDIT)	Section 47.2.187
0x0D2C + *n**0x40	Q*n* Interrupt Control Status register (MTL_Q*n*_INTERRUPT_CONTROL_STATUS)	Section 47.2.188
0x0D30 + *n**0x40	Receive Q*n* Operation Mode register (MTL_RXQ*n*_OPERATION_MODE)	Section 47.2.189
0x0D34 + *n**0x40	Receive Q*n* Missed Packet and Overflow Counter register (MTL_RXQ*n*_MISSED_PACKET_OVERFLOW_CNT)	Section 47.2.190
0x0D38 + *n**0x40	Receive Q*n* Debug register (MTL_RXQ*n*_DEBUG)	Section 47.2.191
0x0D3C + *n**0x40	Receive Q*n* Control register (MTL_RXQ*n*_CONTROL)	Section 47.2.192
0x0D80–0x0FFF	Reserved	

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
DMA registers		
0x1000	DMA_Mode register (DMA_MODE)	Section 47.2.193
0x1004	System Bus Mode register (DMA_SYSBUS_MODE)	Section 47.2.194
0x1008	Interrupt Status register (DMA_INTERRUPT_STATUS)	Section 47.2.195
0x100C	Debug Status 0 register (DMA_DEBUG_STATUS0)	Section 47.2.196
DMA_CHn registers (for *n* = 0 to 1)		
0x1100 + *n**0x80	DMA Channel *n* Control register (DMA_CH*n*_CONTROL)	Section 47.2.197
0x1104 + *n**0x80	DMA Channel *n* Transmit Control register (DMA_CH*n*_TX_CONTROL)	Section 47.2.198
0x1108 + *n**0x80	DMA Channel *n* Receive Control register (DMA_CH*n*_RX_CONTROL)	Section 47.2.199
0x1114 + *n**0x80	DMA Channel *n* Transmit Descriptor List Address register (DMA_CH*n*_TXDESC_LIST_ADDRESS)	Section 47.2.200
0x111C + *n**0x80	DMA Channel *n* Receive Descriptor List Address register(DMA_CH*n*_RXDESC_LIST_ADDRESS)	Section 47.2.201
0x1120 + *n**0x80	DMA Channel *n* Transmit Descriptor Tail Pointer register(DMA_CH*n*_TXDESC_TAIL_POINTER)	Section 47.2.202
0x1128 + *n**0x80	DMA Channel *n* Receive Descriptor Tail Pointer (DMA_CH*n*_RXDESC_TAIL_POINTER)	Section 47.2.203
0x112C + *n**0x80	DMA Channel *n* Transmit Descriptor Ring Length register (DMA_CH*n*_TXDESC_RING_LENGTH)	Section 47.2.204
0x1130 + *n**0x80	DMA Channel *n* Receive Descriptor Ring Length register (DMA_CH*n*_RXDESC_RING_LENGTH)	Section 47.2.205
0x1134 + *n**0x80	DMA Channel *n* Interrupt Enable register (DMA_CH*n*_INTERRUPT_ENABLE)	Section 47.2.206
0x1138 + *n**0x80	DMA Channel *n* Receive Interrupt Watchdog Timer register (DMA_CH*n*_Rx_INTERRUPT_WATCHDOG_TIMER)	Section 47.2.207
0x113C + *n**0x8	DMA Channel *n* Slot Function Control Status register (DMA_CH*n*_SLOT_FUNCTION_CONTROL_STATUS)	Section 47.2.208
0x1144 + *n**0x8	DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_TXDESC)	Section 47.2.209
0x114C + *n**0x8	DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_RXDESC)	Section 47.2.210
0x1154 + *n**0x8	DMA Channel *n* Current Application Transmit Buffer register (DMA_CH*n*_CURRENT_APP_TXBUFFER)	Section 47.2.211
0x115C + *n**0x8	DMA Channel *n* Current Application Receive Buffer register (DMA_CH*n*_CURRENT_APP_RXBUFFER)	Section 47.2.212

Table 654. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x1160 + *n**0x8	DMA Channel *n* Status register (DMA_CH*n*_STATUS)	*Section 47.2.213*
0x116C + *n**0x8	DMA Channel *n* Miss Frame Counter register (DMA_CH*n*_MISS_FRAME_CNT)	*Section 47.2.214*

47.2.1 MAC Configuration Register (MAC_CONFIGURATION)

The MAC Configuration Register establishes the operating mode of the MAC.

Offset 0x0000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ARPEN	SARC			IPC	IPG			GPSLCE	S2KP	CST	ACS	WD	0	JD	JE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	0	BL		DC	PRELEN		TE	RE
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 594. MAC Configuration Register (MAC_CONFIGURATION)

Table 655. MAC_CONFIGURATION field descriptions

Field	Description
0 ARPEN	ARP Offload Enable When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It will forward the ARP packet to the application and also indicate the events in the RxStatus. When this bit is reset, the receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.
1:3 SARC	Source Address Insertion or Replacement Control This field controls the source address insertion or replacement for all transmitted packets. Bit 1 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[2:3]: 0x If TDES3[SAIC] = 3'b000 then the application provides the correct SA field else insertion logic will work according to the setting of TDES3[SAIC]. 10 If Bit 1 is set to 0, the MAC inserts the content of the MAC Address 0 registers (MAC registers 192 and 193) in the SA field of all transmitted packets. If Bit 1 is set to 1 and the Enable MAC Address Register 1 option is selected while configuring the core, the MAC inserts the content of the MAC Address 1 registers (MAC registers 194 and 195) in the SA field of all transmitted packets. 11 If Bit 1 is set to 0, the MAC replaces the content of the MAC Address 0 registers (MAC registers 192 and 193) in the SA field of all transmitted packets. If Bit 1 is set to 1 and the MAC Address Register 1 is enabled, the MAC replaces the content of the MAC Address 1 registers (MAC registers 194 and 195) in the SA field of all transmitted packets. **Note:** Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value. These bits are reserved and RO when the Enable SA and VLAN Insertion on Tx feature is not selected while configuring the core.
4 IPC	Checksum Offload When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.
5:7 IPG	Inter-Packet Gap These bits control the minimum IPG between packets during transmission. 000 96 bit times 001 88 bit times 010 80 bit times ... 111 40 bit times This range of minimum IPG is valid in full-duplex mode. In the half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered. When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG. The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in MAC_Ext_Configuration register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in MAC_Ext_Configuration register.

Table 655. MAC_CONFIGURATION field descriptions (continued)

Field	Description
8 GPSLCE	Giant Packet Size Limit Control Enable When this bit is set, the MAC considers the value in GPSL field in MAC_Ext_Configuration register to declare a received packet as Giant packet. This field must be programed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit. When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet). The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programed giant packet limit should be less than the watchdog limit to get the giant packet status.
9 S2KP	IEEE 802.3 as Support for 2K Packets When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets. When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, refer to Giant Packet Status based on S2KP and JE Bits. **Note:** When the JE bit is set, setting this bit has no effect on the giant packet status.
10 CST	CRC stripping for Type packets When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled. **Note:** For information about how the settings of the ACS bit and this bit impact the packet length, refer to *Table 656*.
11 ACS	Automatic Pad or CRC Stripping When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When this bit is reset, the MAC passes all incoming packets to the application, without any modification. For more details, refer to *Table 656* **Note:** For information about how the settings of CST bit and this bit impact the packet length, refer to *Table 656*.
12 WD	Watchdog Disable When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes. When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Table 655. MAC_CONFIGURATION field descriptions (continued)

Field	Description
14 JD	Jabber Disable When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes. When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.
15 JE	Jumbo Packet Enable When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.
16 PS	Port Select 1 For 10 or 100 Mbps operations
17 FES	Speed This bit selects the speed in the 10/100 Mbps mode: 0 10 Mbps 1 100 Mbps
18 DM	Duplex Mode When this bit is set, the MAC operates in the full-duplex mode in which it can transmit and receive simultaneously.
19 LM	Loopback Mode When this bit is set, the MAC operates in the loopback mode at MII. The MII Rx clock input (clk_rx_i) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.
20 ECRSFD	Enable Carrier Sense Before Transmission in Full-Duplex Mode When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the full-duplex mode. The MAC starts the transmission only when the CRS signal is low. When this bit is reset, the MAC transmitter ignores the status of the CRS signal.
21 DO	Disable Receive Own When this bit is set, the MAC disables the reception of packets when the MII_TX_EN is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY. This bit is not applicable in the full-duplex mode.
22 DCRS	Disable Carrier Sense During Transmission When this bit is set, the MAC transmitter ignores the MII CRS signal during packet transmission in the half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission. When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.
23 DR	Disable Retry When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status. When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the half-duplex mode.

Table 655. MAC_CONFIGURATION field descriptions (continued)

Field	Description
25:26 BL	Back-Off Limit The back-off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. 00 k = min (n, 10) 01 k = min (n, 8) 10 k = min (n, 4) 11 k = min (n, 1) where - = retransmission attempt The random integer r takes the value in the range 0 <= r < 2^k This bit is applicable only in the half-duplex mode.
27 DC	Deferral Check When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on MII. The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in the half-duplex mode.
28:29 PRELEN	Preamble Length for Transmit packets These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the full-duplex mode. 00 7 bytes of preamble 01 5 bytes of preamble 10 3 bytes of preamble 11 Reserved
30 TE	Transmitter Enable When this bit is set, the Tx state machine of the MAC is enabled for transmission on the MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.
31 RE	Receiver Enable When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the MII interface.

Table 656. Packet length based on the CST and ACS bits

Receive checksum offload engine	Receive packet length	CST	ACS	FCS stripping done
IPCKSUM_EN = 0 and IPC_FULL_OFFLOAD = 0 or IPCKSUM_EN = 1 and IPC_FULL_OFFLOAD = 1	< 1536	x	0	No
		x	1	Yes (for Ethernet packets)
	>= 1536	0	x	No
		1	x	Yes (for Type packets)
IPCKSUM_EN = 1 and IPC_FULL_OFFLOAD = 0	< 1536	x	0	No
		x	1	Yes (for Ethernet packets)
	>= 1536	x	x	No

47.2.2 MAC Extended Configuration Register (MAC_EXT_CONFIGURATION)

The MAC Extended Configuration Register establishes the operating mode of the MAC.

Offset 0x0004 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	EIPG					EIPGEN	0	0	0	0	0	USP	SPEN	DCRCC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	GPSL													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 595. MAC_Ext_Configuration Register (MAC_EXT_CONFIGURATION)

Table 657. MAC_EXT_CONFIGURATION field descriptions

Field	Description
2:6 EIPG	Extended Inter-Packet Gap The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits), along with IPG field in MAC_Configuration register, gives the minimum IPG greater than 96 bit times in steps of 8 bit times: {EIPG, IPG} 8'h00 - 104 bit times 8'h01 - 112 bit times 8'h02 - 120 bit times ---------------------- 8'hFF - 2144 bit times
7 EIPGEN	Extended Inter-Packet Gap Enable When this bit is set, the MAC interprets EIPG field and IPG field in MAC_Configuration register together as minimum IPG greater than 96 bit times in steps of 8 bit times. When this bit is reset, the MAC ignores EIPG field and interprets IPG field in MAC_Configuration register as minimum IPG less than or equal to 96 bit times in steps of 8 bit times. **Note:** The extended Inter-Packet Gap feature must be enabled when operating in Full-Duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-Duplex mode.
13 USP	Unicast Slow Protocol Packet Detect When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the MAC_Address0_High and MAC_Address0_Low registers. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02). When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.
14 SPEN	Slow Protocol Detection Enable When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid sub-types. When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.
15 DCRCC	Disable CRC Checking for Received Packets When this bit is set, the MAC receiver does not check the CRC field in the received packets. When this bit is reset, the MAC receiver always checks the CRC field in the received packets.
18:31 GPSL	Giant Packet Size Limit If the received packet size is greater than the value programed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programed in this field must be greater than or equal to 1,518 bytes. Any other programed value is considered as 1,518 bytes. For VLAN tagged packets, the MAC adds 4 bytes to the programed value. When the Enable Double VLAN Processing option is selected, the MAC adds 8 bytes to the programed value for double VLAN tagged packets. The value in this field is applicable when the GPSLCE bit is set in MAC_Configuration register.

47.2.3 MAC Packet Filter Register (MAC_PACKET_FILTER)

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

Offset 0x0008 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RA	0	0	0	0	0	0	0	0	0	DNTU	IPFE	0	0	0	VTFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	HPF	SAF	SAIF	PCF		DBF	PM	DAIF	HMC	HUC	PR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 596. MAC Packet Filter Register (MAC_PACKET_FILTER)

Table 658. MAC_PACKET_FILTER field descriptions

Field	Description
0 RA	Receive All When this bit is set, the MAC Receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word. When this bit is reset, the Receiver module passes only those packets to the application that pass the SA or DA address filter.
10 DNTU	Drop Non-TCP/UDP over IP Packets When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.
11 IPFE	Layer 3 and Layer 4 Filter Enable When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect. When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.
15 VTFE	VLAN Tag Filter Enable When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.
21 HPF	Hash or Perfect Filter When this bit is set, the address filter passes a packet if it matches either the perfect filtering or hash filtering as set by the HMC or HUC bit. When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Table 658. MAC_PACKET_FILTER field descriptions (continued)

Field	Description
22 SAF	Source Address Filter Enable When this bit is set, the MAC compares the SA field of the received packets with the values programed in the enabled SA registers. If the comparison fails, the MAC drops the packet. When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison. **Note:** According to the IEEE specification, Bit 47 of the SA is reserved. However, in the Ethernet module, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.
23 SAIF	SA Inverse Filtering When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programed in the SA registers, it is marked as failing the SA Address filter. When this bit is reset, if the SA of a packet does not match the values programed in the SA registers, it is marked as failing the SA Address filter.
24:25 PCF	Pass Control Packets These bits control the forwarding of all control packets (including unicast and multicast Pause packets). 00 The MAC filters all control packets from reaching the application. 01 The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter. 10 The MAC forwards all control packets to the application even if they fail the Address filter. 11 The MAC forwards the control packets that pass the Address filter.
26 DBF	Disable Broadcast Packets When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings. When this bit is reset, the AFM module passes all received broadcast packets.
27 PM	Pass All Multicast When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.
28 DAIF	DA Inverse Filtering When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.
29 HMC	Hash Multicast When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the hash table. When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programed in DA registers.

Table 658. MAC_PACKET_FILTER field descriptions (continued)

Field	Description
30 HUC	Hash Unicast When this bit is set, the MAC performs the destination address filtering of unicast packets according to the hash table. When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programed in DA registers.
31 PR	Promiscuous Mode When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

47.2.4 Watchdog Timeout Register (MAC_WATCHDOG_TIMEOUT)

The Watchdog Timeout register controls the watchdog timeout for received packets.

Offset 0x000C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	PWE	0	0	0	0	WTO			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 597. MAC Watchdog Timeout Register (MAC_WATCHDOG_TIMEOUT)

Table 659. MAC_WATCHDOG_TIMEOUT field descriptions

Field	Description
23 PWE	Programmable Watchdog Enable When this bit is set and the WD bit of the MAC_Configuration register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in MAC_Configuration register.
28:31 WTO	Watchdog Timeout When the PWE bit is set and the WD bit of the MAC_Configuration register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet. Encoding is as follows: 0x0 2 KB 0x1 3 KB 0x2 4 KB 0x3 5 KB ... 0xC 14 KB 0xD 15 KB 0xE 16383 Bytes 0xF Reserved **Note:** When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

47.2.5 Hash Table Register 0 (MAC_HASH_TABLE_REG0)

The Hash Table Register 0 contains the first 32 bits of the hash table.

The Hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six (seven in 128-bit Hash or eight in 256-bit Hash) bits of the CRC register are used to index the content of the Hash table. The most significant bit determines the register to be used (Hash Table Register X), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects Bit 0 of the Hash Table Register 1.

The hash value of the destination address is calculated in the following way:

- Calculate the 32-bit CRC for the DA (Refer to IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
- Perform bitwise reversal for the value obtained in Step 1.
- Take the upper 6 (or 7 or 8) bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in MAC_Packet_Filter, all multicast packets are accepted regardless of the multicast hash values.

If the Hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written.

If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Offset 0x0010 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HT31T0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HT31T0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 598. MAC Hash Table Reg0 Register (MAC_HASH_TABLE_REG0)

Table 660. MAC_HASH_TABLE_REG0 field descriptions

Field	Description
0:31 HT31T0	This field contains the first 32 Bits [0:31] of the Hash table.

47.2.6 Hash Table Register 1 (MAC_HASH_TABLE_REG1)

The Hash Table Register 1 contains the second 32 bits of the hash table.

The Hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six (seven in 128-bit Hash or eight in 256-bit Hash) bits of the CRC register are used to index the content of the Hash table. The most significant bits determine the register to be used (Hash Table Register X), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects Bit 0 of the Hash Table Register 1.

The hash value of the destination address is calculated in the following way:

- Calculate the 32-bit CRC for the DA (refer to IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
- Perform bitwise reversal for the value obtained in Step 1.
- Take the upper 6 (or 7 or 8) bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in MAC_Packet_Filter, all multicast packets are accepted regardless of the multicast hash values.

If the Hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written.

If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Offset 0x0014 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	HT63T32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	HT63T32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 599. Hash Table Reg1 Register (MAC_HASH_TABLE_REG1)

Table 661. MAC_HASH_TABLE_REG1 field descriptions

Field	Description
0:31 HT63T32	This field contains the second 32 Bits [31:0] of the Hash table.

47.2.7 VLAN Tag Register (MAC_VLAN_TAG)

The VLAN Tag register identifies the IEEE 802.1Q VLAN type packets.

Offset 0x0050 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIVLRXS	0	EIVLS		ERIVLT	EDVLP	VTHM	EVLRXS	0	EVLS		DOVLTC	ERSVLM	ESVL	VTIM	ETV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	VL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 600. VLAN Tag Register (MAC_VLAN_TAG)

Table 662. MAC_VLAN_TAG field descriptions

Field	Description
0 EIVLRXS	Enable Inner VLAN Tag in Rx Status When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.
2:3 EIVLS	Enable Inner VLAN Tag Stripping on Receive This field indicates the stripping operation on inner VLAN Tag in received packet: 00 Do not strip 01 Strip if VLAN filter passes 10 Strip if VLAN filter fails 11 Always strip
4 ERIVLT	Enable Inner VLAN Tag When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.
5 EDVLP	Enable Double VLAN Processing When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).
6 VTHM	VLAN Tag Hash Table Match Enable When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the MAC_VLAN_Hash_Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN hash table. When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison. When this bit is reset, the VLAN Hash Match operation is not performed.
7 EVLRXS	Enable VLAN Tag in Rx status When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.
9:10 EVLS	Enable VLAN Tag Stripping on Receive This field indicates the stripping operation on the outer VLAN Tag in received packet: 00 Do not strip 01 Strip if VLAN filter passes 10 Strip if VLAN filter fails 11 Always strip
11 DOVLTC	Disable VLAN Type Check When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN. When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Table 662. MAC_VLAN_TAG field descriptions (continued)

Field	Description
12 ERSVLM	Enable Receive S-VLAN Match When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets. The ERIVLT bit determines the VLAN tag position considered for filtering or matching.
13 ESVL	Enable S-VLAN When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.
14 VTIM	VLAN Tag Inverse Match Enable When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.
15 ETV	Enable 12-Bit VLAN Tag Comparison When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[20:31] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for hash-based VLAN filtering. When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN hash filtering.
16:31 VL	VLAN Tag Identifier for Receive Packets This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field: – Bits[16:18]: User Priority – Bit 19: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) – Bits[20:31]: VLAN Identifier (VID) field of VLAN tag When the ETV bit is set, only the VID is used for comparison. If this field ([20:31] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.

47.2.8 VLAN Hash Table Register (MAC_VLAN_HASH_TABLE)

When the ERSVLM bit of MAC_Hash_Table_Reg1 register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of MAC_VLAN_Tag Register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table.

The hash value of the destination address is calculated in the following way:

- Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, refer to Section 3.2.8 of IEEE 802.3 - 2008).
- Perform bitwise reversal for the value obtained in step 1.
- Take the upper four bits from the value obtained in step 2.

If the VLAN hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[15:8] (in little-endian mode) or Bits[7:0] (in big-endian mode) of this register are written.

- If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Offset 0x0058 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	VLHT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 601. VLAN_Hash_Table Register (MAC_VLAN_HASH_TABLE)

Table 663. MAC_VLAN_HASH_TABLE field descriptions

Field	Description
16:31 VLHT	VLAN Hash Table This field contains the 16-bit VLAN Hash Table.

47.2.9 VLAN Tag Inclusion Register (MAC_VLAN_INCL)

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

Offset 0x0060 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	VLTI	CSVL	VLP	VLC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	VLT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 602. VLAN Tag Inclusion Register (MAC_VLAN_INCL)

Table 664. MAC_VLAN_INCL field descriptions

Field	Description
11 VLTI	VLAN Tag Input When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken the Tx descriptor.
12 CSVL	C-VLAN or S-VLAN When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.
13 VLP	VLAN Priority Control When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.
14:15 VLC	VLAN Tag Control in Transmit Packets 00 No VLAN tag deletion, insertion, or replacement 01 VLAN tag deletion, The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags. 10 VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 11 VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8). **Note:** Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.
16:31 VLT	VLAN Tag for Transmit Packets This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[16:18] are the User Priority field, Bit 19 is the CFI/DEI field, and Bits[20:31] are the VID field in the VLAN tag. The following list describes the bits of this field: – Bits[16:18] User Priority – Bit 19 Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) – Bits[20:31] VLAN Identifier (VID) field of VLAN tag

47.2.10 Inner VLAN Tag Inclusion Register (MAC_INNER_VLAN_INCL)

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

Offset 0x0064 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	VLTI	CSVL	VLP	VLC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	VLT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 603. Inner VLAN Tag Inclusion Register (MAC_INNER_VLAN_INCL)

Table 665. MAC_INNER_VLAN_INCL field descriptions

Name	Description
11 VLTI	VLAN Tag Input When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.
12 CSVL	C-VLAN or S-VLAN When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.
13 VLP	VLAN Priority Control When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.
14:15 VLC	VLAN Tag Control in Transmit Packets 00 No VLAN tag deletion, insertion, or replacement 01 VLAN tag deletion. The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags. 10 VLAN tag insertion. The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 11 VLAN tag replacement. The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8). **Note:** Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.
16:31 VLT	VLAN Tag for Transmit Packets This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[16:18] are the User Priority field, Bit 19 is the CFI/DEI field, and Bits[20:31] are the VID field in the VLAN tag. The following list describes the bits of this field: – Bits[16:18] User Priority – Bit 19 Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) – Bits[20:31] VLAN Identifier (VID) field of VLAN tag

47.2.11 MAC Q0 Flow Control Register (MAC_Q0_TX_FLOW_CTRL)

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

Offset 0x0070 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	PT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DZPQ	PLT			0	0	TFE	FCB_BPA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 604. MAC Q0 Flow Control Register (MAC_Q0_TX_FLOW_CTRL)

Table 666. MAC_Q0_TX_FLOW_CTRL field descriptions

Name	Description
0:15 PT	Pause Time This field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.
24 DZPQ	Disable Zero-Quanta Pause When this bit is set, it disables the automatic generation of the zero-quanta Pause packets on de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i). When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.

Table 666. MAC_Q0_TX_FLOW_CTRL field descriptions (continued)

Name	Description
25:27 PLT	Pause Low Threshold This field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the Pause packet. The threshold values should be always less than the Pause Time configured in Bits[0:15]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first Pause packet is transmitted. The following list provides the threshold values for different values: 000 Pause Time minus 4 Slot Times (PT -4 slot times) 001 Pause Time minus 28 Slot Times (PT -28 slot times) 010 Pause Time minus 36 Slot Times (PT -36 slot times) 011 Pause Time minus 144 Slot Times (PT -144 slot times) 100 Pause Time minus 256 Slot Times (PT -256 slot times) 101 Pause Time minus 512 Slot Times (PT -512 slot times) 110 Reserved 111 Reserved The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.
30 TFE	Transmit Flow Control Enable Full-Duplex Mode: In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets. Half-Duplex Mode: In the half-duplex mode, when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.
31 FCB_BPA	Flow Control Busy or Backpressure Activate This bit initiates a Pause packet in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. Full-Duplex Mode: In the full-duplex mode, this bit should be read as 1'b0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1'b1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 1'b0. You should not write to this register until this bit is cleared. Half-Duplex Mode: When this bit is set (and TFE bit is set) in the half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.12 MAC Q1 Flow Control Register (MAC_Q1_TX_FLOW_CTRL)

This register controls the generation of PFC Control packets of priorities mapped as per the PSRQ*i* field in the MAC_RxQ_Ctrl2 registers. This register is present only when the Enable Data Center Bridging feature is selected during configuration.

Offset 0x0074 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DZPQ	PLT			0	0	TFE	FCB_BPA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 605. MAC Q1 Flow Control Register (MAC_Q1_TX_FLOW_CTRL)

Table 667. MAC_Q1_TX_FLOW_CTRL field descriptions

Name	Description
0:15 PT	Pause Time This field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.
24 DZPQ	Disable Zero-Quanta Pause When this bit is set, it disables the automatic generation of the zero-quanta Pause packets on de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i). When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.
25:27 PLT	Pause Low Threshold This field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the Pause packet. The threshold values should be always less than the Pause Time configured in Bits[0:15]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first Pause packet is transmitted. The following list provides the threshold values for different values: 000 Pause Time minus 4 Slot Times (PT -4 slot times) 001 Pause Time minus 28 Slot Times (PT -28 slot times) 010 Pause Time minus 36 Slot Times (PT -36 slot times) 011 Pause Time minus 144 Slot Times (PT -144 slot times) 100 Pause Time minus 256 Slot Times (PT -256 slot times) 101 Pause Time minus 512 Slot Times (PT -512 slot times) 110 Reserved 111 Reserved The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Table 667. MAC_Q1_TX_FLOW_CTRL field descriptions (continued)

Name	Description
30 TFE	When this bit is set in full-duplex mode, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.
31 FCB_BPA	This bit initiates a PFC packet if the TFE bit is set. To initiate a PFC packet, the application must set this bit to 1'b1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When PFC packet transmission is complete, the MAC resets this bit to 1'b0. You should not write to this register until this bit is cleared. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.13 MAC Receive Flow Control Register (MAC_RX_FLOW_CTRL)

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

Offset 0x0090 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	UP	RFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 606. MAC Receive Flow Control Register (MAC_RX_FLOW_CTRL)

Table 668. MAC_RX_FLOW_CTRL field descriptions

Field	Description
30 UP	Unicast Pause Packet Detect A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in MAC_Address0_High and MAC_Address0_Low. When this bit is reset, the MAC only detects Pause packets with unique multicast address. **Note:** The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011.
31 RFE	Receive Flow Control Enable When this bit is set and the MAC is operating in full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in half-duplex mode, the decode function of the Pause packet is disabled. When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.

47.2.14 Transmit Queue Priority Mapping 0 Register (MAC_TXQ_PRTY_MAP0)

The Transmit Queue Priority Mapping 0 register contains the priority values assigned to Tx Queue 0 and Tx Queue1.

Offset 0x0098 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	PSTQ1								PSTQ0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 607. Transmit Queue Priority Mapping 0 Register (MAC_TXQ_PRTY_MAP0)

Table 669. MAC_TXQ_PRTY_MAP0 field descriptions

Field	Description
16:23 PSTQ1	Priorities Selected in Transmit Queue 1 This bit is similar to the PSTQ0 bit.
24:31 PSTQ0	Priorities Selected in Transmit Queue 0 This field holds the priorities assigned to Tx Queue 0 by the software. This field determines if Tx Queue 0 should be blocked from transmitting specified pause time when a PFC packet is received with priorities matching the priorities programed in this field. If the content of this field is not mutually exclusive to corresponding fields of other Transmit queues, that is, same priority is mapped to multiple Tx queues, the MAC blocks all queues with matching priority for specified time.

47.2.15 Receive Queue Control 0 Register (MAC_RXQ_CTRL0)

The Receive Queue Control 0 register controls the queue management in the MAC Receiver.

Offset 0x00A0 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXQ1EN		RXQ0EN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 608. Receive Queue Control 0 Register (MAC_RXQ_CTRL0)

Table 670. MAC_RXQ_CTRL0 field descriptions

Field	Description
28:29 RXQ1EN	Receive Queue 1 Enable This field indicates whether Rx Queue 1 is enabled for audio video (AV) or data center bridging (DCB). 00 Not enabled 01 Queue 1 enabled for AV 10 Queue 1 enabled for DCB or generic 11 Reserved
30:31 RXQ0EN	Receive Queue 0 Enable This field indicates whether Rx Queue 0 is enabled for audio video (AV) or data center bridging (DCB). 00 Not enabled 01 Queue 0 enabled for AV 10 Queue 0 enabled for DCB or generic 11 Reserved

47.2.16 Receive Queue Control 1 Register (MAC_RXQ_CTRL1)

The Receive Queue Control 1 register controls the routing of multicast, broadcast, AV, and untagged packets to the Rx queues.

Offset 0x00A4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	MCBCQEN	0	MCBCQ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	UPQ			0	0	0	0	0	AVPTPQ			0	AVCPQ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 609. Receive Queue Control 1 Register (MAC_RXQ_CTRL1)

Table 671. MAC_RXQ_CTRL1 field descriptions

Field	Description
11 MCBCQEN	Multicast and Broadcast Queue Enable This bit specifies that Multicast or Broadcast packets routing to the Rx Queue is enabled and the Multicast or Broadcast packets must be routed to Rx Queue specified in MCBCQ field.
13:15 MCBCQ	Multicast and Broadcast Queue This field specifies the Rx Queue onto which Multicast or Broadcast Packets are routed. Any Rx Queue enabled for Generic/AV traffic can be used to route the Multicast or Broadcast Packets. 000 Rx Queue 0 001 Rx Queue 1 others Reserved
17:19 UPQ	Untagged Packet Queue This field indicates the Rx Queue to which Untagged Packets are to be routed. Any Rx Queue enabled for Generic/AV traffic can be used to route the Untagged Packets. 000 Rx Queue 0 001 Rx Queue 1 others Reserved

Table 671. MAC_RXQ_CTRL1 field descriptions (continued)

Field	Description
25:27 AVPTPQ	AV PTP Packets Queue This field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed. 000 Rx Queue 0 001 Rx Queue 1 others: Reserved When the AV8021ASMEN bit of MAC_Timestamp_Control register is set, only untagged PTP over Ethernet packets are routed on an Rx queue.
29:31 AVCPQ	AV Untagged Control Packets Queue This field specifies the Receive queue on which the received AV untagged control packets are routed: 000 Receive Queue 0 001 Receive Queue 1 others: Reserved The AV tagged control and data packets are routed based on PSRQ field in the Transmit Flow Control Register of corresponding queue.

47.2.17 Receive Queue Control 2 Register (MAC_RXQ_CTRL2)

This register controls the routing of tagged packets based on the USP (user Priority) field of the received packets to the RxQueues 0 to 1.

Offset 0x00A8 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	PSRQ1								PSRQ0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 610. Receive Queue Control 2 Register (MAC_RXQ_CTRL2)

Table 672. MAC_RXQ_CTRL2 field descriptions

Name	Description
16:23 PSRQ1	Priorities Selected in the Receive Queue 1 This field decides the priorities assigned to Rx Queue 1. All packets with priorities that match the values set in this field are routed to Rx Queue 1. For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx Queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.
24:31 PSRQ0	Priorities Selected in the Receive Queue 0 This field decides the priorities assigned to Rx Queue 0. All packets with priorities that match the values set in this field are routed to Rx Queue 0. For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx Queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. Value After Reset: 0x0

47.2.18 Interrupt Status Register (MAC_INTERRUPT_STATUS)

The Interrupt Status register contains the status of interrupts.

Offset 0x00B0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	RXSTSIS	TXSTSIS	TSIS	MMCRXIPIS	MMCTXIS	MMCRXIS	MMCIS	0	0	LPIIS	PMTIS	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 611. Interrupt Status Register (MAC_INTERRUPT_STATUS)

Table 673. MAC_INTERRUPT_STATUS field descriptions

Field	Description
17 RXSTSIS	Receive Status Interrupt This bit indicates the status of received packets. This bit is set when the RWT bit is set in the MAC_Rx_Tx_Status register. This bit is cleared when the corresponding interrupt source bit is read in the MAC_Rx_Tx_Status register.
18 TXSTSIS	Transmit Status Interrupt This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the MAC_Rx_Tx_Status register: – Excessive Collision (EXCOL) – Late Collision (LCOL) – Excessive Deferral (EXDEF) – Loss of Carrier (LCARR) – No Carrier (NCARR) – Jabber Timeout (TJT) This bit is cleared when the corresponding interrupt source bit is read in the MAC_Rx_Tx_Status register.
19 TSIS	Timestamp Interrupt Status This bit is set when any of the following conditions is true: – The system time value is equal to or exceeds the value specified in the Target Time High and Low registers. – There is an overflow in the Seconds register. – The Target Time Error occurred, that is, programed target time already elapsed. – When the auxiliary snapshot trigger is asserted. – When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the MAC_TxTimestamp_Status_Nanoseconds and Mac_TxTimestamp_Status_Seconds registers. This bit is cleared when the corresponding interrupt source bit is read in the MAC_Timestamp_Status register.
20 MMCRXIPIS	MMC Receive Checksum Offload Interrupt Status This bit is set high when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared.
21 MMCTXIS	MMC Transmit Interrupt Status This bit is set high when an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared.
22 MMCRXIS	MMC Receive Interrupt Status This bit is set high when an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared.
23 MMCIS	MMC Interrupt Status This bit is set high when Bit 20, Bit 21, or Bit 22 is set high. This bit is cleared only when all these bits are low.

Table 673. MAC_INTERRUPT_STATUS field descriptions (continued)

Field	Description
26 LPIIS	LPI Interrupt Status This bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared when the TLPIEN bit of MAC_LPI_Control_Status register is read.
27 PMTIS	PMT Interrupt Status This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in MAC_PMT_Control_Status register). This bit is cleared when Bits[25:26] are cleared because of a Read operation to the MAC_PMT_Control_Status register.

47.2.19 Interrupt Enable Register (MAC_INTERRUPT_ENABLE)

The Interrupt Enable register contains the masks for generating the interrupts.

Offset 0x00B4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	RXSTSIE	TXSTSIE	TSIE	0	0	0	0	0	0	LPIIE	PMTIE	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 612. Interrupt Enable Register (MAC_INTERRUPT_ENABLE)

Table 674. MAC_INTERRUPT_ENABLE field descriptions

Field	Description
17 RXSTSIE	Receive Status Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the MAC_Interrupt_Status register.
18 TXSTSIE	Transmit Status Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the MAC_Interrupt_Status register.
19 TSIE	Timestamp Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in MAC_Interrupt_Status register.

Table 674. MAC_INTERRUPT_ENABLE field descriptions (continued)

Field	Description
26 LPIIE	LPI Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in MAC_Interrupt_Status register.
27 PMTIE	PMT Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in MAC_Interrupt_Status register.

47.2.20 Receive Transmit Status Register (MAC_RX_TX_STATUS)

The Receive Transmit Status register contains the Receive and Transmit Error status.

Offset 0x00B8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RWT	0	0	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 613. Receive Transmit Status Register (MAC_RX_TX_STATUS)

Table 675. MAC_RX_TX_STATUS field descriptions

Field	Description
23 RWT	Receive Watchdog Timeout This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the MAC_Configuration register. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the MAC_Configuration register. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 EXCOL	Excessive Collisions When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after the first collision and the packet transmission is aborted. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 675. MAC_RX_TX_STATUS field descriptions (continued)

Field	Description
27 LCOL	Late Collision When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode). This bit is not valid if the Underflow error occurs. Access restriction applies. Clears on read. Self-set to 1 on internal event.
28 EXDEF	Excessive Deferral When the DTXSTS bit is set in the MTL_Operation_Mode register and the DC bit is set in the MAC_Configuration register, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 when Jumbo packet is enabled). Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 LCARR	Loss of Carrier When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the loss of carrier occurred during packet transmission, that is, the phy_crs_i signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision. Access restriction applies. Clears on read. Self-set to 1 on internal event.
30 NCARR	No Carrier When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 TJT	Transmit Jabber Timeout This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the MAC_Configuration register. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the MAC_Configuration register. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.21 PMT Control and Status Register (MAC_PMT_CONTROL_STATUS)

This register captures the remote wakeup control and status.

Offset 0x00C0 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RWKFILTRST	0	0	0	0	RWKPTR			0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RWKPFE	GLBLUCAST	0	0	RWKPRCVD	MGKPRCVD	0	0	RWKPKTEN	MGKPKTEN	PWRDWN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 614. PMT Control and Status Register (MAC_PMT_CONTROL_STATUS)

Table 676. MAC_PMT_CONTROL_STATUS field descriptions

Field	Description
0 RWKFILTRST	Remote Wake-Up Packet Filter Register Pointer Reset When this bit is set, the remote wake-up packet filter register pointer is reset to 3'b000. It is automatically cleared after 1 clock cycle. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
5:7 RWKPTR	Remote Wake-up FIFO Pointer This field gives the current value (0 to 7) of the Remote Wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote Wake-up Packet Filter Register are transferred to the clk_rx_i domain when a Write occurs to that register.
21 RWKPFE	Remote Wake-up Packet Forwarding Enable When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected Wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet. The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high. **Note:** If Magic Packet Enable and Wake-Up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
22 GLBLUCAST	Global Unicast When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Table 676. MAC_PMT_CONTROL_STATUS field descriptions (continued)

Field	Description
25 RWKPRCVD	Remote Wake-Up Packet Received When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 MGKPRCVD	Magic Packet Received When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 RWKPKTEN	Remote Wake-Up Packet Enable When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.
30 MGKPKTEN	Magic Packet Enable When this bit is set, a power management event is generated when the MAC receives a magic packet.
31 PWRDWN	Power Down When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote Wake-Up Packet Enable bit is set high. **Note:** You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.22 Wake-up Packet Filter Registers

The wkuppktfilter_reg register at address 0x00C4 loads the Wake-up Packet Filter register.

To load values in a Wake-up Packet Filter register, the entire register (wkuppktfilter_reg) must be written. The wkuppktfilter_reg register is loaded by sequentially loading the eight register values in address (0x00C4) for wkuppktfilter_reg0, wkuppktfilter_reg1, …, wkuppktfilter_reg7, respectively. The wkuppktfilter_reg register is read in a similar way.

The Ethernet module updates the wkuppktfilter_reg register current pointer value in Bits[5:7] of MAC_PMT_Control_Status register.

- Filter i Byte Mask: The filter i byte mask register defines the bytes of the packet that are examined by filter i (0, 1, 2, 3) to determine whether or not a packet is a wake-up packet.
  - The MSB (bit 0) must be zero.
  - Bit j[1:31] is the byte mask.
  - If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter i Offset + j of the incoming packet; otherwise Filter i Offset + j is ignored.
- Filter i Command: The 4-bit filter i command controls the filter i operation.
  - Bit 28 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet.
  - Bit 29 (Inverse Mode), when set, reverses the logic of the CRC16 hash function signal, to reject a packet with matching CRC_16 value.
  - Bit 29, along with Bit 30, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2".
  - Bit 30 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set.
  - Bit 31 is the enable for filter i. If Bit 31 is not set, filter i is disabled.
- Filter i Offset: This filter i offset register defines the offset (within the packet) from which the filter i examines the packets.
  - This 8-bit pattern-offset is the offset for the filter i first byte to be examined.
  - The minimum allowed offset is 12, which refers to the 13th byte of the packet.
  - The offset value 0 refers to the first byte of the packet.
- Filter i CRC-16: This filter i CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programed to the wake-up filter register block.
  - The 16-bit CRC calculation uses the following polynomial:
    $G(x) = x^{16} + x^{15} + x^2 + 1$
    Each mask, used in the hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following:
  - 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation.
  - 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation.

The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

Note: If you are accessing these registers in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when CPU accesses Lane 3 (or Lane 0 in big-endian mode).

When any Register content is being transferred to a different clock domain after a write operation, there should not be any further writes to the same location until the first write is updated. Otherwise, the second write operation does not get updated to the destination clock domain. Therefore, the delay between two writes to the same register location should *be at least 4 cycles of the destination clock (PHY receive clock, PHY transmit clock, or PTP clock).*

Table 677. Remote Wake-Up Packet Filter Register

wkuppktfilter_reg0	Filter 0 Byte Mask							
wkuppktfilter_reg1	Filter 1 Byte Mask							
wkuppktfilter_reg2	Filter 2 Byte Mask							
wkuppktfilter_reg3	Filter 3 Byte Mask							
wkuppktfilter_reg4	RSVD	Filter 3 Command	RSVD	Filter 2 Command	RSVD	Filter 1 Command	RSVD	Filter 0 Command
wkuppktfilter_reg5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset	
wkuppktfilter_reg6	Filter 1 CRC - 16				Filter 0 CRC - 16			
wkuppktfilter_reg7	Filter 3 CRC - 16				Filter 2 CRC - 16			

47.2.22.1 Wakeup Filter0 Byte Mask Register (RWK_FILTER0_BYTE_MASK)

Corresponds to registers wkuppktfilter_reg0 described in *Table 677*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Filter0_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Filter0_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 615. Wakeup Filter0 Byte Mask Register (RWK_FILTER0_BYTE_MASK)

Table 678. RWK_FILTER0_BYTE_MASK field descriptions

Field	Description
0:31 Filter0_Byte_Mask	The filter 0 byte mask register defines the bytes of the packet that are examined by filter 0 to determine whether or not a packet is a wake-up packet. The MSB (bit 0) must be zero. Bit j[1:31] is the byte mask. If bit j (byte number) of the byte mask is set, the CRC block processes the filter 0 offset + j of the incoming packet; otherwise filter 0 offset + j is ignored.

47.2.22.2 Wakeup Filter 1 Byte Mask Register (RWK_FILTER1_BYTE_MASK)

Corresponds to registers wkuppktfilter_reg1 described in *Table 677*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Filter1_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	Filter1_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 616. Wakeup Filter 1 Byte Mask Register (RWK_FILTER1_BYTE_MASK)

Table 679. RWK_FILTER1_BYTE_MASK field descriptions

Field	Description
0:31 Filter1_Byte_Mask	The filter 1 byte mask register defines the bytes of the packet that are examined by filter 1 to determine whether or not a packet is a wake-up packet. The MSB (bit 0) must be zero. Bit j[1:31] is the byte mask. If bit j (byte number) of the byte mask is set, the CRC block processes the filter 1 offset + j of the incoming packet; otherwise filter 1 offset + j is ignored.

47.2.22.3 Wakeup Filter 2 Byte Mask Register (RWK_FILTER2_BYTE_MASK)

Corresponds to registers wkuppktfilter_reg2 described in *Table 677*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Filter2_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	Filter2_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 617. Wakeup Filter 2 Byte Mask Register (RWK_FILTER2_BYTE_MASK)

Table 680. RWK_FILTER2_BYTE_MASK field descriptions

Field	Description
0:31 Filter2_Byte_Mask	The filter 2 byte mask register defines the bytes of the packet that are examined by filter 2 to determine whether or not a packet is a wake-up packet. The MSB (bit 0) must be zero. Bit j[1:31] is the byte mask. If bit j (byte number) of the byte mask is set, the CRC block processes the filter 2 offset + j of the incoming packet; otherwise filter 2 offset + j is ignored.

47.2.22.4 Wakeup Filter 3 Byte Mask Register (RWK_FILTER3_BYTE_MASK)

Corresponds to registers wkuppktfilter_reg3 described in *Table 677*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Filter3_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Filter3_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 618. Wakeup Filter 3 Byte Mask Register (RWK_FILTER3_BYTE_MASK)

Table 681. RWK_FILTER3_BYTE_MASK field descriptions

Field	Description
0:31 Filter3_Byte_Mask	The filter 3 byte mask register defines the bytes of the packet that are examined by filter 3 to determine whether or not a packet is a wake-up packet. The MSB (bit 0) must be zero. Bit j[1:31] is the byte mask. If bit j (byte number) of the byte mask is set, the CRC block processes the filter 3 offset + j of the incoming packet; otherwise filter 3 offset + j is ignored.

47.2.22.5 Wakeup Filter 0123 Command Register (RWK_FILTER0123_COMMAND)

Corresponds to registers wkuppktfilter_reg4 described in *Table 677*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	Filter3_Command				0	0	0	0	Filter2_Command			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	Filter1_Command				0	0	0	0	Filter0_Command			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 619. Wakeup Filter 0123 Command Register (RWK_FILTER0123_COMMAND)

Table 682. RWK_FILTER0123_COMMAND field descriptions

Field	Description
4:7 Filter3_Command	Refer to the description in *Section 47.2.22*.
12:15 Filter2_Command	Refer to the description in *Section 47.2.22*.
20:23 Filter1_Command	Refer to the description of *Section 47.2.22*.
28:31 Filter0_Command	Refer to the description of *Section 47.2.22*.

47.2.22.6 Wakeup Filter 0123 Offset Register (RWK_FILTER0123_OFFSET)

Corresponds to registers wkuppktfilter_reg5 described in *Table 677*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Filter3_Offset								Filter2_Offset							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	Filter1_Offset								Filter0_Offset							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 620. Wakeup Filter 0123 Offset Register (RWK_FILTER0123_OFFSET)

Table 683. RWK_FILTER0123_OFFSET field descriptions

Field	Description
0:7 Filter3_Offset	Refer to the description of *Section 47.2.22*.
8:15 Filter2_Offset	Refer to the description of *Section 47.2.22*.
16:23 Filter1_Offset	Refer to the description of *Section 47.2.22*.
24:31 Filter0_Offset	Refer to the description of *Section 47.2.22*.

47.2.22.7 Wakeup Filter01 CRC Register (RWK_FILTER01_CRC)

Corresponds to registers wkuppktfilter_reg6 described in *Table 677*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Filter1_CRC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Filter0_CRC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 621. Wakeup Filter01 CRC Register (RWK_FILTER01_CRC)

Table 684. RWK_FILTER01_CRC field descriptions

Field	Description
0:15 Filter1_CRC	Refer to the description in *Section 47.2.22*.
16:31 Filter0_CRC	Refer to the description in *Section 47.2.22*.

47.2.22.8 Wakeup Filter23 CRC Register (RWK_FILTER23_CRC)

Corresponds to registers wkuppktfilter_reg7 described in *Table 677*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Filter3_CRC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	Filter2_CRC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 622. Wakeup Filter23 CRC Register (RWK_FILTER23_CRC)

Table 685. RWK_FILTER23_CRC field descriptions

Field	Description
0:15 Filter3_CRC	Refer to the description of *Section 47.2.22*.
16:31 Filter2_CRC	Refer to the description of *Section 47.2.22*.

47.2.23 LPI Control and Status Register (MAC_LPI_CONTROL_STATUS)

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

Offset 0x00D0 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	LPITCSE	LPIATE	LPITXA	0	PLS	LPIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	RLPIST	TLPIST	0	0	0	0	RLPIEX	RLPIEN	TLPIEX	TLPIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 623. LPI Control and Status Register (MAC_LPI_CONTROL_STATUS)

Table 686. MAC_LPI_CONTROL_STATUS field descriptions

Field	Description
10 LPITCSE	LPI Tx Clock Stop Enable When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode.
11 LPIATE	LPI Timer Enable This bit controls the automatic entry of the MAC Transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPITXEN bits are set, the MAC Transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the MAC_LPI_Entry_Timer register. After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the Transmitter exits LPI state but does not clear LPITXEN bit. This enables the re-entry into LPI state when it is IDLE again. When LPITE is 0, the LPI Auto timer is disabled and MAC Transmitter enters LPI state based on the settings of LPITXA and LPITXEN bit descriptions.
12 LPITXA	LPI Tx Automate This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side. If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of the MTL_TxQ0_Operation_Mode register, when the MAC is in the LPI mode, it exits the LPI mode. When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.
14 PLS	PHY Link Status This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER. When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down. Software should get the PHY link status by reading the PHY register and accordingly update the PLS bit
15 LPIEN	LPI Enable When this bit is set, it instructs the MAC Transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission. This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.
22 RLPIST	Receive LPI State When this bit is set, it indicates that the MAC is receiving the LPI pattern on the MII interface.
23 TLPIST	Transmit LPI State When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.

Table 686. MAC_LPI_CONTROL_STATUS field descriptions (continued)

Field	Description
28 RLPIEX	Receive LPI Exit When this bit is set, it indicates that the MAC Receiver has stopped receiving the LPI pattern on the MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register. **Note:** This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.
29 RLPIEN	Receive LPI Entry When this bit is set, it indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register. **Note:** This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.
30 TLPIEX	Transmit LPI Exit When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register.
31 TLPIEN	Transmit LPI Entry When this bit is set, it indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register.

47.2.24 LPI Timers Control Register (MAC_LPI_TIMERS_CONTROL)

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

Offset 0x00D4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	LST									
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TWT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 624. LPI Timers Control Register (MAC_LPI_TIMERS_CONTROL)

Table 687. MAC_LPI_TIMERS_CONTROL field descriptions

Field	Description
6:15 LST	LPI LS TIMER This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.
16:31 TWT	LPI TW TIMER This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

47.2.25 LPI Entry Timer Register (MAC_LPI_ENTRY_TIMER)

This register controls the Tx LPI entry timer. This counter is enabled only when bit[11](LPITE) bit of MAC_LPI_Control_Status is set to 1.

Offset 0x00D8 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	LPIET			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPIET													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 625. LPI Entry Timer Register (MAC_LPI_ENTRY_TIMER)

Table 688. MAC_LPI_ENTRY_TIMER field descriptions

Field	Description
12:28 LPIET	LPI Entry Timer This field specifies the time in microseconds the MAC will wait to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1. Bits [29:31] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

47.2.26 1US Tic Counter Register (MAC_1US_TIC_COUNTER)

This register controls the generation of the Reference time (1 microsecond tic) for all the LPI timers. This timer has to be programed by the software initially.

Offset 0x00DC Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TIC_1US_CNTR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 626. 1US Tic Counter Register (MAC_1US_TIC_COUNTER)

Table 689. MAC_1US_TIC_COUNTER field descriptions

Field	Description
20:31 TIC_1US_CNTR	1US TIC Counter The application must program this counter so that the number of clock cycles of CSR clock is 1us. (Subtract 1 from the value before programming). For example if the CSR clock is 100MHz then this field needs to be programed to value 100 - 1 = 99 (which is 0x63). This is required to generate the 1US events that are used to update some of the EEE related counters.

47.2.27 Version Register (MAC_VERSION)

The version register identifies the version of the Ethernet module.

Offset 0x0110 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	USERVER								VENDORVER							
W																
Reset	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1

Figure 627. Version Register (MAC_VERSION)

Table 690. MAC_VERSION field descriptions

Field	Description
16:23 USERVER	User IP Version Number
24:31 VENDORVER	Vendor IP Version Number

47.2.28 Debug Register (MAC_DEBUG)

The Debug register provides the debug status of various MAC blocks.

Offset 0x0114 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TFCSTS		TPESTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	RFCFCSTS		RPESTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 628. Debug Register (MAC_DEBUG)

Table 691. MAC_DEBUG field descriptions

Field	Description
13:14 TFCSTS	MAC Transmit Packet Controller Status This field indicates the state of the MAC Transmit Packet Controller module: 00 Idle state 01 Waiting for one of the following: Status of the previous packet OR IPG or backoff period to be over 10 Generating and transmitting a Pause control packet (in full-duplex mode) 11 Transferring input packet for transmission
15 TPESTS	MAC MII Transmit Protocol Engine Status When this bit is set, it indicates that the MAC MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.
29:30 RFCFCSTS	MAC Receive Packet Controller FIFO Status When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.
31 RPESTS	MAC MII Receive Protocol Engine Status When this bit is set, it indicates that the MAC MII receive protocol engine is actively receiving data, and it is not in the Idle state.

47.2.29 Hardware Feature0 Register (MAC_HW_FEATURE0)

This register indicates the presence of the optional features or functions of the Ethernet module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Offset 0x011C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	ACTPHYSEL			SAVLANINS	TSSTSSEL		MACADR64SEL	MACADR32SEL	ADDMACADRSEL					0	RXCOESEL
W																
Reset	0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TXCOESEL	EEESEL	TSSEL	0	0	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	0	HDSEL	0	MIISEL
W																
Reset	0	1	1	1	0	0	1	1	1	1	1	1	0	1	0	1

Figure 629. Hardware Feature0 Register (MAC_HW_FEATURE0)

Table 692. MAC_HW_FEATURE0 field descriptions

Field	Description
1:3 ACTPHYSEL	Active PHY Selected When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion: 000 MII 100 RMII All Others Reserved
4 SAVLANINS	Source Address or VLAN Insertion Enable
5:6 TSSTSSEL	Timestamp System Time Source This bit indicates the source of the Timestamp system time: 01 Internal
7 MACADR64SEL	MAC Addresses 64-127 Selected This bit is set to 1 when the Additional 64 MAC Address Registers (64-127) are present.
8 MACADR32SEL	MAC Addresses 32-63 Selected This bit is set to 1 when the Additional 32 MAC Address Registers (32-63) are present.
9:13 ADDMACADRSEL	MAC Addresses 1-31 Selected This bit is set to 1 when the Additional 1-31 MAC Address Registers are present
15 RXCOESEL	Receive Checksum Offload present This bit is set to 1 when the Receive TCP/IP Checksum Check is present

Table 692. MAC_HW_FEATURE0 field descriptions (continued)

Field	Description
17 TXCOESEL	Transmit Checksum Offload present
18 EEESEL	Energy Efficient Ethernet present
19 TSSEL	IEEE 1588-2008 Timestamp present
22 ARPOFFSEL	ARP Offload present
23 MMCSEL	RMON Module Present
24 MGKSEL	PMT Magic Packet present
25 RWKSEL	PMT Remote Wake-up Packet Enable
26 SMASEL	SMA (MDIO) Interface present
27 VLHASH	VLAN Hash Filter enabled
29 HDSEL	Half-duplex Support
31 MIISEL	10 or 100 Mbps Support

47.2.30 Hardware Feature1 Register (MAC_HW_FEATURE1)

This register indicates the presence of the optional features or functions of the Ethernet module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Offset 0x0120 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	L3L4FNUM				0	HASHTBLSZ		LPMODEEN		0	AVSEL	DBGMEMA	0	0	0
W																
Reset	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR64		ADVTHWORD	0	OSTEN	TXFIFOSIZE					0	RXFIFOSIZE				
W																
Reset	0	0	1	0	1	0	0	1	0	1	0	0	0	1	1	0

Figure 630. Hardware Feature1 Register (MAC_HW_FEATURE1)

Table 693. MAC_HW_FEATURE1 field descriptions

Field	Description
1:4 L3L4FNUM	Total number of L3 or L4 Filters This field indicates the total number of L3 or L4 filters: 0001 1 L3 or L4 Filter
6:7 HASHTBLSZ	Hash Table Size This field indicates the size of the hash table: 01 64 bit hash table
8 LPMODEEN	Low Power Mode Enabled This bit is set to 1 when the Enable UPF-Based Low-Power Support is present.
11 AVSEL	AV Feature Enabled This bit is set to 1 when the Enable Audio Video Bridging is present.
12 DBGMEMA	DMA Debug Registers Enabled This bit is set to 1 when the Debug Mode is present
16:17 ADDR64	Address Width. This field indicates the configured address width: 00 32-bit address width
18 ADVTHWORD	IEEE 1588 High Word Register Present
20 OSTEN	One-Step Timestamping Present

Table 693. MAC_HW_FEATURE1 field descriptions (continued)

Field	Description
21:25 TXFIFOSIZE	MTL Transmit FIFO Size This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7: 00101 4,096 bytes
27:31 RXFIFOSIZE	MTL Receive FIFO Size This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7: 00110 8,192 bytes

47.2.31 Hardware Feature2 Register (MAC_HW_FEATURE2)

This register indicates the presence of the optional features or functions of the Ethernet module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Offset 0x0124 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	AUXSNAPNUM			0	PPSOUTNUM			0	0	TXCHCNT				0	0
W																
Reset	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXCHCNT				0	0	TXQCNT				0	0	RXQCNT			
W																
Reset	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1

Figure 631. Hardware Feature2 Register (MAC_HW_FEATURE2)

Table 694. MAC_HW_FEATURE2 field descriptions

Field	Description
1:3 AUXSNAPNUM	Number of Auxiliary Snapshot Inputs This field indicates the number of auxiliary snapshot inputs: 010 2 auxiliary inputs others Reserved
5:7 PPSOUTNUM	Number of PPS Outputs This field indicates the number of PPS outputs: 001 1 PPS output others Reserved
10:13 TXCHCNT	Number of DMA Transmit Channels This field indicates the number of DMA Transmit channels: 0001 2 DMA Tx Channels
16:19 RXCHCNT	Number of DMA Receive Channels This field indicates the number of DMA Receive channels: 0001 2 DMA Rx Channels

Table 694. MAC_HW_FEATURE2 field descriptions (continued)

Field	Description
22:25 TXQCNT	Number of MTL Transmit Queues This field indicates the number of MTL Transmit queues: 0001 2 MTL Tx Queues
28:31 RXQCNT	Number of MTL Receive Queues This field indicates the number of MTL Receive queues: 0001 2 MTL Rx Queues

47.2.32 MDIO Address Register (MAC_MDIO_ADDRESS)

The MDIO Address register controls the management cycles to external PHY through a management interface.

Offset 0x0200 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PSE	BTB	PA					RDA				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NTC			CR				0	0	0	SKAP	GOC		C45E	GB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 632. MDIO Address Register (MAC_MDIO_ADDRESS)

Table 695. MAC_MDIO_ADDRESS field description

Field	Description
4 PSE	Preamble Suppression Enable When this bit is set, the SMA will suppress the 32-bit preamble and transmit MDIO frames with only 1 preamble bit. When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.
5 BTB	Back to Back transactions When this bit is set and the NTC has value greater than 0, then the MAC will inform the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which will be executed immediately irrespective of the number trailing clocks generated for the previous frame. When this bit is reset, then the read/write command completion (GB is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame. This bit must not be set when NTC=0.
6:10 PA	Physical Layer Address This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing. This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

Table 695. MAC_MDIO_ADDRESS field description (continued)

Field	Description
11:15 RDA	Register/Device Address These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.
17:19 NTC	Number of Training Clocks This field controls the number of trailing clock cycles generated on sma_mdc_o (MDC) after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 3'h3 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.
20:23 CR	CSR Clock Range The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency used in your design: 0000 CSR clock = 60-100 MHz; MDC clock = CSR clock/42 0001 CSR clock = 100-150 MHz; MDC clock = CSR clock/62 0010 CSR clock = 20-35 MHz; MDC clock = CSR clock/16 0011 CSR clock = 35-60 MHz; MDC clock = CSR clock/26 0100 CSR clock = 150-250 MHz; MDC clock = CSR clock/102 0101 CSR clock = 250-300 MHz; MDC clock = CSR clock/124 0110 Reserved 0111 Reserved The suggested range of CSR clock frequency applicable for each value (when Bit 20 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When Bit 20 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits as 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks: 1000 CSR clock/4 1001 CSR clock/6 1010 CSR clock/8 1011 CSR clock/10 1100 CSR clock/12 1101 CSR clock/14 1110 CSR clock/16 1111 CSR clock/18
27 SKAP	Skip Address Packet When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.
28:29 GOC	PHY SMA interface Operation Command This field indicates the operation command to the PHY: 00 Reserved 01 Write 10 Post Read Increment Address for Clause 45 PHY 11 Read When Clause 22 PHY is enabled, only Write (01) and Read (11) commands are valid.

Table 695. MAC_MDIO_ADDRESS field description (continued)

Field	Description
30 C45E	Clause 45 PHY Enable When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.
31 GB	PHY SMA interface Busy The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIO slave. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in MAC_MDIO_Address and MAC_MDIO_Data registers as long as this bit is set. For write transfers, the application must first write 16-bit data in the GDI field (and also RA field when C45E is set) in MAC_MDIO_Data register before setting this bit. When C45E is set, it should also write into the RA field of MAC_MDIO_Data register before initiating a read transfer. When a read transfer is completed (GB=0), the data read from the PHY register is valid in the GD field of the MAC_MDIO_Data register. **Note:** Even if the addressed PHY is not present, there is no change in the functionality of this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.33 MDIO Data Register (MAC_MDIO_DATA)

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in MAC_MDIO_Address. This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

Offset 0x0204 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 633. MDIO Data Register (MAC_MDIO_DATA)

Table 696. MAC_MDIO_DATA field descriptions

Name	Description
0:15 RA	Register Address This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.
16:31 GD	PHY management Data This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

47.2.34 ARP Address Register (MAC_ARP_ADDRESS)

The ARP Address register contains the IPv4 Destination Address of the MAC.

Offset 0x0210 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ARPPA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ARPPA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 634. ARP Address Register (MAC_ARP_ADDRESS)

Table 697. MAC_ARP_ADDRESS field descriptions

Name	Description
0:31 ARPPA	ARP Protocol Address This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

47.2.35 Address0 High Register (MAC_ADDRESS0_HIGH)

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the MII interface corresponds to the LS byte (Bits [24:31]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the MacAddress0 Register [0:47] is compared with 0x665544332211.

Offset 0x0300 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCS
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ADDRHI															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 635. Address0 High Register (MAC_ADDRESS0_HIGH)

Table 698. MAC_ADDRESS0_HIGH field descriptions

Field	Description
0 AE	Address Enable This bit is always set to 1.
15 DCS	DMA Channel Select This field contains the DMA Channel number to which the Rx packet whose DA matches the MAC Address0 content is routed. 0 Route RX packet to DMA channel0 1 Route RX packet to DMA channel1
16:31 ADDRHI	MAC Address0[0:15] This field contains the upper 16 bits [0:15] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

47.2.36 Address0 Low Register (MAC_ADDRESS0_LOW)

The MAC Address0 Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

Offset 0x0304 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ADDRLO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ADDRLO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 636. Address0 Low Register (MAC_ADDRESS0_LOW)

Table 699. MAC_ADDRESS0_LOW field descriptions

Name	Description
0:31 ADDRLO	MAC Address0[16:47] This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

47.2.37 MAC Address High register 1 to 31 (MAC_ADDRESS*n*_HIGH31_1)

The MAC Address1 High register holds the upper 16 bits of the second 6-byte MAC address of the station.

Offset 0x0300 + *n**0x8 (n=1 to 31) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	AE	SA	MBC						0	0	0	0	0	0	0	DCS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ADDRHI															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 637. MAC Address High register 1 to 31 (MAC_ADDRESS*n*_HIGH31_1)

Table 700. MAC_ADDRESS*n*_HIGH31_1 field descriptions

Field	Description
0 AE	Address Enable When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
1 SA	Source Address When this bit is set, the MAC Address1[0:47] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address1[0:47] is used to compare with the DA fields of the received packet.
2:7 MBC	Mask Byte Control These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: Bit 2: Register 194[16:23] Bit 3: Register 194[24:31] Bit 4: Register 195[0:7] … Bit 7: Register 195[24:31] You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.
15 DCS	DMA Channel Select This field contains the DMA Channel number to which the Rx packet whose DA matches the MAC Address1 content is routed. 0 Route RX packet to DMA channel0 1 Route RX packet to DMA channel1
16:31 ADDRHI	MAC Address1 [0:15] This field contains the upper 16 bits[0:15] of the second 6-byte MAC address.

47.2.38 MAC Address Low register 1 to 31 (MAC_ADDRESSn_LOW31_1)

The MAC Address1 Low register holds the lower 32 bits of the second 6-byte MAC address of the station.

Offset 0x0304 + *n**0x8 (n=1 to 31) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ADDRLO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ADDRLO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 638. MAC Address Low register 1 to 31 (MAC_ADDRESSn_LOW31_1)

Table 701. MAC_ADDRESSn_LOW31_1 field descriptions

Field	Description
0:31 ADDRLO	MAC Address1 [31:0] This field contains the lower 32 bits of second 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process.

47.2.39 MAC Address High register 32 to 63 (MAC_ADDRESSn_HIGH63_32)

The MAC Address32 High register holds the upper 16 bits of the 33rd 6-byte MAC address of the station.

Offset 0x0300 + *n**0x8 (n=32 to 63) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ADDRHI															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 639. MAC Address High register 32 to 63 (MAC_ADDRESSn_HIGH63_32)

Table 702. MAC_ADDRESS*n*_HIGH63_32 field descriptions

Field	Description
0 AE	Address Enable When this bit is set, the Address filter module uses the 33rd MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
15 DCS	DMA Channel Select This field contains the DMA Channel number to which an Rx packet whose DA matches the MAC Address32 content is routed. 0 Route RX packet to DMA channel0 1 Route RX packet to DMA channel1
16:31 ADDRHI	MAC Address32 [0:15] This field contains the upper 16 bits (0:15) of the 33rd 6-byte MAC address.

47.2.40 MAC Address Low register 32 to 63 (MAC_ADDRESS*n*_LOW63_32)

The MAC Address32 Low register holds the lower 16 bits of the 33rd 6-byte MAC address of the station.

Offset 0x0304 + *n**0x8 (n=32 to 63) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDRLO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDRLO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 640. MAC Address Low register 32 to 63 (MAC_ADDRESS*n*_LOW63_32)

Table 703. MAC_ADDRESS*n*_LOW63_32 field descriptions

Field	Description
0:31 ADDRLO	MAC Address32 [16:47] This field contains the lower 32 bits of the 33rd 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process

47.2.41 MAC Address High register 64 to 127 (MAC_ADDRESS*n*_HIGH127_64)

The MAC Address32 High register holds the upper 16 bits of the 33rd 6-byte MAC address of the station.

Offset 0x0300 + *n**0x8 (n=64 to 127) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ADDRHI															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 641. MAC Address High register 64 to 127 (MAC_ADDRESS*n*_HIGH127_64)

Table 704. MAC_ADDRESS*n*_HIGH127_64 field descriptions

Field	Description
0 AE	Address Enable When this bit is set, the Address filter module uses the 33rd MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
15 DCS	DMA Channel Select This field contains the DMA Channel number to which an Rx packet whose DA matches the MAC Address32 content is routed. 0 Route RX packet to DMA channel0 1 Route RX packet to DMA channel1
16:31 ADDRHI	MAC Address32 [0:15] This field contains the upper 16 bits (0:15) of the 33rd 6-byte MAC address.

47.2.42 MAC Address Low register 64 to 127 (MAC_ADDRESS*n*_LOW127_64)

The MAC Address32 Low register holds the lower 16 bits of the 33rd 6-byte MAC address of the station.

Offset 0x0304 + *n**0x8 (n=64 to 127) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ADDRLO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ADDRLO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 642. MAC Address Low register 64 to 127 (MAC_ADDRESS*n*_LOW127_64)

Table 705. MAC_ADDRESSn_LOW127_64 field descriptions

Field	Description
0:31 ADDRLO	MAC Address32 [16:47] This field contains the lower 32 bits of the 33rd 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process

47.2.43 MMC Control Register (MMC_CONTROL)

The MMC Control register establishes the operating mode of the management counters.

Offset 0x0700 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	UCDBC	0	0	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 643. MMC Control Register (MMC_CONTROL)

Table 706. MMC_CONTROL field descriptions

Field	Description
23 UCDBC	Update MMC Counters for Dropped Broadcast Packets When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of MAC_Packet_Filter register. When reset, the MMC Counters are not updated for dropped Broadcast packets.
26 CNTPRSTLVL	Full-Half Preset When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (Half 2KBytes) and all packet-counters gets preset to 0x7FFF_FFF0 (Half 16). When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (Full 2KBytes) and all packet-counters gets preset to 0xFFFF_FFF0 (Full 16).

Table 706. MMC_CONTROL field descriptions (continued)

Field	Description
27 CNTPRST	Counters Preset When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle. This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets. The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.
28 CNTFREEZ	MMC Counter Freeze When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.
29 RSTONRD	Reset on Read When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[0:7]) is read.
30 CNTSTOPRO	Counter Stop Rollover When this bit is set, the counter does not roll over to zero after reaching the maximum value.
31 CNTRST	Counters Reset When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets. The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

47.2.44 MMC Receive Interrupt Register (MMC_RX_INTERRUPT)

The MMC Receive Interrupt register maintains the interrupts that are generated when the following happens:

- Receive statistic counters reach half of their maximum values, that is 0x8000_0000.
- Receive statistic counters cross their maximum values, that is 0xFFFF_FFFF.
- When the Counter Stop Rollover is set, interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[0:7]) of the respective counter must be read in order to clear the interrupt bit.

Offset 0x0704 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	RXLPITRCIS	RXLPIUSCIS	RXCTRLPIS	RXRCVERRPIS	RXWDOGPIS	RXVLANGBPIS	RXFOVPIS	RXPAUSPIS	RXORANGEPIS	RXLENERPIS	RXUCGPIS	RX1024TMAXOCTGBPIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX512T1023OCTGBPIS	RX256T511OCTGBPIS	RX128T255OCTGBPIS	RX65T127OCTGBPIS	RX64OCTGBPIS	RXOSIZEGPIS	RXUSIZEGPIS	RXJABERPIS	RXRUNTPIS	RXALGNERPIS	RXCRCERPIS	RXMCGPIS	RXBCGPIS	RXGOCTIS	RXGBOCTIS	RXGBPKTIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 644. MMC Receive Interrupt Register (MMC_RX_INTERRUPT)

Table 707. MMC_RX_INTERRUPT field descriptions

Field	Description
4 RXLPITRCIS	MMC Receive LPI Transition Counter Interrupt Status This bit is set when the RX_LPI_TRAN_CNTR counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5 RXLPIUSCIS	MMC Receive LPI Microsecond Counter Interrupt Status This bit is set when the RX_LPI_USEC_CNTR counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
6 RXCTRLPIS	MMC Receive Control Packet Counter Interrupt Status This bit is set when the rxctrlpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 707. MMC_RX_INTERRUPT field descriptions (continued)

Field	Description
7 RXRCVERRPIS	MMC Receive Error Packet Counter Interrupt Status This bit is set when the rxrcverror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
8 RXWDOGPIS	MMC Receive Watchdog Error Packet Counter Interrupt Status This bit is set when the rxwatchdog error counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
9 RXVLANGBPIS	MMC Receive VLAN Good Bad Packet Counter Interrupt Status This bit is set when the rxvlanpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
10 RXFOVPIS	MMC Receive FIFO Overflow Packet Counter Interrupt Status This bit is set when the rxfifooverflow counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
11 RXPAUSPIS	MMC Receive Pause Packet Counter Interrupt Status This bit is set when the rxpausepackets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
12 RXORANGEPIS	MMC Receive Out Of Range Error Packet Counter Interrupt Status. This bit is set when the rxoutofrangetype counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
13 RXLENERPIS	MMC Receive Length Error Packet Counter Interrupt Status This bit is set when the rxlengtherror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
14 RXUCGPIS	MMC Receive Unicast Good Packet Counter Interrupt Status This bit is set when the rxunicastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
15 RX1024TMAXOCTGBPIS	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 707. MMC_RX_INTERRUPT field descriptions (continued)

Field	Description
16 RX512T1023OCTGBPIS	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
17 RX256T511OCTGBPIS	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
18 RX128T255OCTGBPIS	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
19 RX65T127OCTGBPIS	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
20 RX64OCTGBPIS	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx64octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21 RXOSIZEGPIS	MMC Receive Oversize Good Packet Counter Interrupt Status This bit is set when the rxoversize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
22 RXUSIZEGPIS	MMC Receive Undersize Good Packet Counter Interrupt Status This bit is set when the rxundersize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
23 RXJABERPIS	MMC Receive Jabber Error Packet Counter Interrupt Status This bit is set when the rxjabbererror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 707. MMC_RX_INTERRUPT field descriptions (continued)

Field	Description
24 RXRUNTPIS	MMC Receive Runt Packet Counter Interrupt Status This bit is set when the rxrunterror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
25 RXALGNERPIS	MMC Receive Alignment Error Packet Counter Interrupt Status This bit is set when the rxalignmenterror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 RXCRCERPIS	MMC Receive CRC Error Packet Counter Interrupt Status This bit is set when the rxcrcerror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
27 RXMCGPIS	MMC Receive Multicast Good Packet Counter Interrupt Status This bit is set when the rxmulticastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
28 RXBCGPIS	MMC Receive Broadcast Good Packet Counter Interrupt Status This bit is set when the rxbroadcastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 RXGOCTIS	MMC Receive Good Octet Counter Interrupt Status This bit is set when the rxoctetcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
30 RXGBOCTIS	MMC Receive Good Bad Octet Counter Interrupt Status This bit is set when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 RXGBPKTIS	MMC Receive Good Bad Packet Counter Interrupt Status This bit is set when the rxpacketcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.45 MMC Tx Interrupt Register (MMC_TX_INTERRUPT)

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values, that is 0x8000_0000, and when they cross their maximum values, that is 0xFFFF_FFFF. When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[0:7]) of the respective counter must be read in order to clear the interrupt bit.

Offset 0x0708 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TXLPITRCIS	TXLPIUSCIS	TXOSIZEGPIS	TXVLANGPIS	TXPAUSPIS	TXEXDEFPIS	TXGPKTIS	TXGOCTIS	TXCARERPIS	TXEXCOLPIS	TXLATCOLPIS	TXDEFPIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXMCOLGPIS	TXSCOLGPIS	TXUFLOWERPIS	TXBCGBPIS	TXMCGBPIS	TXUCGBPIS	TX1024TMAXOCTGBPIS	TX512T1023OCTGBPIS	TX256T511OCTGBPIS	TX128T255OCTGBPIS	TX65T127OCTGBPIS	TX64OCTGBPIS	TXMCGPIS	TXBCGPIS	TXGBPKTIS	TXGBOCTIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 645. MMC Tx Interrupt Register (MMC_TX_INTERRUPT)

Table 708. MMC_TX_INTERRUPT field descriptions

Field	Description
4 TXLPITRCIS	MMC Transmit LPI Transition Counter Interrupt Status This bit is set when the TX_LPI_TRAN_CNTR counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5 TXLPIUSCIS	MMC Transmit LPI Microsecond Counter Interrupt Status This bit is set when the TX_LPI_USEC_CNTR counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
6 TXOSIZEGPIS	MMC Transmit Oversize Good Packet Counter Interrupt Status This bit is set when the txoversize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 708. MMC_TX_INTERRUPT field descriptions (continued)

Field	Description
7 TXVLANGPIS	MMC Transmit VLAN Good Packet Counter Interrupt Status This bit is set when the txvlanpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
8 TXPAUSPIS	MMC Transmit Pause Packet Counter Interrupt Status This bit is set when the txpausepacketserror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
9 TXEXDEFPIS	MMC Transmit Excessive Deferral Packet Counter Interrupt Status This bit is set when the txexcessdef counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
10 TXGPKTIS	MMC Transmit Good Packet Counter Interrupt Status This bit is set when the txpacketcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
11 TXGOCTIS	MMC Transmit Good Octet Counter Interrupt Status This bit is set when the txoctetcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
12 TXCARERPIS	MMC Transmit Carrier Error Packet Counter Interrupt Status This bit is set when the txcarriererror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
13 TXEXCOLPIS	MMC Transmit Excessive Collision Packet Counter Interrupt Status This bit is set when the txexesscol counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
14 TXLATCOLPIS	MMC Transmit Late Collision Packet Counter Interrupt Status This bit is set when the txlatecol counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
15 TXDEFPIS	MMC Transmit Deferred Packet Counter Interrupt Status This bit is set when the txdeferred counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 708. MMC_TX_INTERRUPT field descriptions (continued)

Field	Description
16 TXMCOLGPIS	MMC Transmit Multiple Collision Good Packet Counter Interrupt Status This bit is set when the txmulticol_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
17 TXSCOLGPIS	MMC Transmit Single Collision Good Packet Counter Interrupt Status This bit is set when the txsinglecol_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
18 TXUFLOWERPIS	MMC Transmit Underflow Error Packet Counter Interrupt Status This bit is set when the txunderflowerror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
19 TXBCGBPIS	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status This bit is set when the txbroadcastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
20 TXMCGBPIS	MMC Transmit Multicast Good Bad Packet Counter Interrupt Status The bit is set when the txmulticastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21 TXUCGBPIS	MMC Transmit Unicast Good Bad Packet Counter Interrupt Status This bit is set when the txunicastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
22 TX1024TMAXOCTGBPIS	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
23 TX512T1023OCTGBPIS	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 708. MMC_TX_INTERRUPT field descriptions (continued)

Field	Description
24 TX256T511OCTGBPIS	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
25 TX128T255OCTGBPIS	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 TX65T127OCTGBPIS	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx65to127octets_gb counter reaches half the maximum value, and also when it reaches the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
27 TX64OCTGBPIS	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx64octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
28 TXMCGPIS	MMC Transmit Multicast Good Packet Counter Interrupt Status This bit is set when the txmulticastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 TXBCGPIS	MMC Transmit Broadcast Good Packet Counter Interrupt Status This bit is set when the txbroadcastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
30 TXGBPKTIS	MMC Transmit Good Bad Packet Counter Interrupt Status This bit is set when the txpacketcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 TXGBOCTIS	MMC Transmit Good Bad Octet Counter Interrupt Status This bit is set when the txoctetcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.46 MMC Rx Interrupt mask Register (MMC_RX_INTERRUPT_MASK)

This register maintains the masks for interrupts generated from all Receive statistics counters.

Offset 0x070C

Access: Software read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	RXLPITRCIM	RXLPIUSCIM	RXCTRLPIM	RXRCVERRPIM	RXWDOGPIM	RXVLANGBPIM	RXFOVPIM	RXPAUSPIM	RXORANGEPIM	RXLENERPIM	RXUCGPIM	RX1024TMAXOCTGBPIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX512T1023OCTGBPIM	RX256T511OCTGBPIM	RX128T255OCTGBPIM	RX65T127OCTGBPIM	RX64OCTGBPIM	RXOSIZEGPIM	RXUSIZEGPIM	RXJABERPIM	RXRUNTPIM	RXALGNERPIM	RXCRCERPIM	RXMCGPIM	RXBCGPIM	RXGOCTIM	RXGBOCTIM	RXGBPKTIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 646. MMC Rx Interrupt mask Register (MMC_RX_INTERRUPT_MASK)

Table 709. MMC_RX_INTERRUPT_MASK field descriptions

Field	Description
4 RXLPITRCIM	MMC Receive LPI Transition Counter Interrupt Mask Setting this bit masks the interrupt when RX_LPI_TRAN_CNTR reaches half the maximum value or the maximum value.
5 RXLPIUSCIM	MMC Receive LPI Microsecond Counter Interrupt Mask Setting this bit masks the interrupt when RX_LPI_USEC_CNTR reaches half the maximum value or the maximum value.
6 RXCTRLPIM	MMC Receive Control Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxctrlpackets_g counter reaches half of the maximum value or the maximum value.
7 RXRCVERRPIM	MMC Receive Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxrcverror counter reaches half of the maximum value or the maximum value.

Table 709. MMC_RX_INTERRUPT_MASK field descriptions (continued)

Field	Description
8 RXWDOGPIM	MMC Receive Watchdog Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxwatchdog counter reaches half of the maximum value or the maximum value.
9 RXVLANGBPIM	MMC Receive VLAN Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxvlanpackets_gb counter reaches half of the maximum value or the maximum value.
10 RXFOVPIM	MMC Receive FIFO Overflow Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxfifooverflow counter reaches half of the maximum value or the maximum value.
11 RXPAUSPIM	MMC Receive Pause Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxpausepackets counter reaches half of the maximum value or the maximum value.
12 RXORANGEPIM	MMC Receive Out Of Range Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoutofrangetype counter reaches half of the maximum value or the maximum value.
13 RXLENERPIM	MMC Receive Length Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxlengtherror counter reaches half of the maximum value or the maximum value.
14 RXUCGPIM	MMC Receive Unicast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxunicastpackets_g counter reaches half of the maximum value or the maximum value.
15 RX1024TMAXOCTGBPIM	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask. Setting this bit masks the interrupt when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.
16 RX512T1023OCTGBPIM	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
17 RX256T511OCTGBPIM	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value.
18 RX128T255OCTGBPIM	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value.
19 RX65T127OCTGBPIM	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value.
20 RX64OCTGBPIM	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx64octets_gb counter reaches half of the maximum value or the maximum value.

Table 709. MMC_RX_INTERRUPT_MASK field descriptions (continued)

Field	Description
21 RXOSIZEGPIM	MMC Receive Oversize Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoversize_g counter reaches half of the maximum value or the maximum value.
22 RXUSIZEGPIM	MMC Receive Undersize Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxundersize_g counter reaches half of the maximum value or the maximum value.
23 RXJABERPIM	MMC Receive Jabber Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxjabbererror counter reaches half of the maximum value or the maximum value.
24 RXRUNTPIM	MMC Receive Runt Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxrunterror counter reaches half of the maximum value or the maximum value.
25 RXALGNERPIM	MMC Receive Alignment Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxalignmenterror counter reaches half of the maximum value or the maximum value.
26 RXCRCERPIM	MMC Receive CRC Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxcrcerror counter reaches half of the maximum value or the maximum value.
27 RXMCGPIM	MMC Receive Multicast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxmulticastpackets_g counter reaches half of the maximum value or the maximum value.
28 RXBCGPIM	MMC Receive Broadcast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxbroadcastpackets_g counter reaches half of the maximum value or the maximum value.
29 RXGOCTIM	MMC Receive Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoctetcount_g counter reaches half of the maximum value or the maximum value.
30 RXGBOCTIM	MMC Receive Good Bad Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value.
31 RXGBPKTIM	MMC Receive Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxpacketcount_gb counter reaches half of the maximum value or the maximum value.

47.2.47 MMC Tx Interrupt Mask Register (MMC_TX_INTERRUPT_MASK)

This register maintains the masks for interrupts generated from all Transmit statistics counters.

Offset 0x0710

Access: Software read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TXLPITRCIM	TXLPIUSCIM	TXOSIZEGPIM	TXVLANGPIM	TXPAUSPIM	TXEXDEFPIM	TXGPKTIM	TXGOCTIM	TXCARERPIM	TXEXCOLPIM	TXLATCOLPIM	TXDEFPIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TXMCOLGPIM	TXSCOLGPIM	TXUFLOWERPIM	TXBCGBPIM	TXMCGBPIM	TXUCGBPIM	TX1024TMAXOCTGBPIM	TX512T1023OCTGBPIM	TX256T511OCTGBPIM	TX128T255OCTGBPIM	TX65T127OCTGBPIM	TX64OCTGBPIM	TXMCGPIM	TXBCGPIM	TXGBPKTIM	TXGBOCTIM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 647. MMC Tx Interrupt Mask Register (MMC_TX_INTERRUPT_MASK)

Table 710. MMC_TX_INTERRUPT_MASK field descriptions

Field	Description
4 TXLPITRCIM	MMC Transmit LPI Transition Counter Interrupt Mask Setting this bit masks the interrupt when TX_LPI_TRAN_CNTR reaches half the maximum value or the maximum value.
5 TXLPIUSCIM	MMC Transmit LPI Microsecond Counter Interrupt Mask Setting this bit masks the interrupt when TX_LPI_USEC_CNTR reaches half the maximum value or the maximum value.
6 TXOSIZEGPIM	MMC Transmit Oversize Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txoversize_g counter reaches half of the maximum value or the maximum value.
7 TXVLANGPIM	MMC Transmit VLAN Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txvlanpackets_g counter reaches half of the maximum value or the maximum value.
8 TXPAUSPIM	MMC Transmit Pause Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txpausepackets counter reaches half of the maximum value or the maximum value.
9 TXEXDEFPIM	MMC Transmit Excessive Deferral Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txexcessdef counter reaches half of the maximum value or the maximum value.
10 TXGPKTIM	MMC Transmit Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txpacketcount_g counter reaches half of the maximum value or the maximum value.

Table 710. MMC_TX_INTERRUPT_MASK field descriptions (continued)

Field	Description
11 TXGOCTIM	MMC Transmit Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the txoctetcount_g counter reaches half of the maximum value or the maximum value.
12 TXCARERPIM	MMC Transmit Carrier Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txcarriererror counter reaches half of the maximum value or the maximum value.
13 TXEXCOLPIM	MMC Transmit Excessive Collision Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txexcesscol counter reaches half of the maximum value or the maximum value.
14 TXLATCOLPIM	MMC Transmit Late Collision Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txlatecol counter reaches half of the maximum value or the maximum value.
15 TXDEFPIM	MMC Transmit Deferred Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txdeferred counter reaches half of the maximum value or the maximum value.
16 TXMCOLGPIM	MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txmulticol_g counter reaches half of the maximum value or the maximum value.
17 TXSCOLGPIM	MMC Transmit Single Collision Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txsinglecol_g counter reaches half of the maximum value or the maximum value.
18 TXUFLOWERPIM	MMC Transmit Underflow Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txunderflowerror counter reaches half of the maximum value or the maximum value.
19 TXBCGBPIM	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txbroadcastpackets_gb counter reaches half of the maximum value or the maximum value.
20 TXMCGBPIM	MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txmulticastpackets_gb counter reaches half of the maximum value or the maximum value.
21 TXUCGBPIM	MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txunicastpackets_gb counter reaches half of the maximum value or the maximum value.
22 TX1024TMAXOCTGBPIM	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.
23 TX512T1023OCTGBPIM	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value.

Table 710. MMC_TX_INTERRUPT_MASK field descriptions (continued)

Field	Description
24 TX256T511OCTGBPIM	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value.
25 TX128T255OCTGBPIM	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value.
26 TX65T127OCTGBPIM	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value.
27 TX64OCTGBPIM	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx64octets_gb counter reaches half of the maximum value or the maximum value.
28 TXMCGPIM	MMC Transmit Multicast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txmulticastpackets_g counter reaches half of the maximum value or the maximum value.
29 TXBCGPIM	MMC Transmit Broadcast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txbroadcastpackets_g counter reaches half of the maximum value or the maximum value.
30 TXGBPKTIM	MMC Transmit Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txpacketcount_gb counter reaches half of the maximum value or the maximum value.
31 TXGBOCTIM	MMC Transmit Good Bad Octet Counter Interrupt Mask Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half of the maximum value or the maximum value.

47.2.48 Transmit Octet Count Good Bad Register (TX_OCTET_COUNT_GOOD_BAD)

This register provides the number of bytes transmitted by the Ethernet module, exclusive of preamble and retried bytes, in good and bad packets.

Offset 0x0714																Access: User Read
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 648. Transmit Octet Count Good Bad Register (TX_OCTET_COUNT_GOOD_BAD)

Table 711. TX_OCTET_COUNT_GOOD_BAD field descriptions

Field	Description
0:31 TXOCTGB	This field indicates the number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad packets.

47.2.49 Transmit Packet Count Good Bad Register (TX_PACKET_COUNT_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module, exclusive of retried packets.

Offset 0x0718																Access: User Read
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 649. Transmit Packet Count Good Bad Register (TX_PACKET_COUNT_GOOD_BAD)

Table 712. TX_PACKET_COUNT_GOOD_BAD field descriptions

Field	Description
0:31 TXPKTGB	This field indicates the number of good and bad packets transmitted, exclusive of retried packets.

47.2.50 Transmit Broadcast Packets Good Register (TX_BROADCAST_PACKETS_GOOD)

This register provides the number of good broadcast packets transmitted by the Ethernet module.

Offset 0x071C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXBCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXBCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 650. Transmit Broadcast Packets Good Register (TX_BROADCAST_PACKETS_GOOD)

Table 713. TX_BROADCAST_PACKETS_GOOD field descriptions

Field	Description
0:31 TXBCASTG	This field indicates the number of good broadcast packets transmitted.

47.2.51 Transmit Multicast Packets Good Register (TX_MULTICAST_PACKETS_GOOD)

This register provides the number of good multicast packets transmitted by the Ethernet module.

Offset 0x0720 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXMCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXMCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 651. Transmit Multicast Packets Good Register (TX_MULTICAST_PACKETS_GOOD)

Table 714. TX_MULTICAST_PACKETS_GOOD field descriptions

Field	Description
0:31 TXMCASTG	This field indicates the number of good multicast packets transmitted.

47.2.52 Transmit 64Octets Packets Good Bad Register (TX_64OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length 64 bytes, exclusive of preamble and retried packets.

Offset 0x0724 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX64OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX64OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 652. Transmit 64Octets Packets Good Bad Register (TX_64OCTETS_PACKETS_GOOD_BAD)

Table 715. TX_64OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX64OCTGB	This field indicates the number of good and bad packets transmitted with length 64 bytes, exclusive of preamble and retried packets.

47.2.53 Transmit 65to127Octets Packets Good Bad Register (TX_65TO127OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x0728 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX65_127OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX65_127OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 653. Transmit 65to127Octets Packets Good Bad Register (TX_65TO127OCTETS_PACKETS_GOOD_BAD)

Table 716. TX_65TO127OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX65_127OCTGB	This field indicates the number of good and bad packets transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

47.2.54 Transmit 128To255Octets Packets Good Bad Register (TX_128TO255OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length between 128 to 255 (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x072C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX128_255OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX128_255OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 654. Transmit 128To255Octets Packets Good Bad Register (TX_128TO255OCTETS_PACKETS_GOOD_BAD)

Table 717. TX_128TO255OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX128_255OCTGB	This field indicates the number of good and bad packets transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried packets.

47.2.55 Transmit 256To511Octets Packets Good Bad Register (TX_256TO511OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length between 256 to 511 (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x0730 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX256_511OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX256_511OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 655. Transmit 256To511Octets Packets Good Bad Register (TX_256TO511OCTETS_PACKETS_GOOD_BAD)

Table 718. TX_256TO511OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX256_511OCTGB	This field indicates the number of good and bad packets transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried packets.

47.2.56 Transmit 512To1023Octets Packets Good Bad Register (TX_512TO1023OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length 512 to 1023 (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x0734 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX512_1023OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX512_1023OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 656. Transmit 512To1023Octets Packets Good Bad Register (TX_512TO1023OCTETS_PACKETS_GOOD_BAD)

Table 719. TX_512TO1023OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX512_1023OCTGB	This field indicates the number of good and bad packets transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried packets.

47.2.57 Transmit 1024ToMaxOctets Packets Good Bad Register (TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length 1024 to maxsize (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x0738 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX1024_MAXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX1024_MAXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 657. Transmit 1024ToMaxOctets Packets Good Bad Register (TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

Table 720. TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX1024_MAXOCTGB	This field indicates the number of good and bad packets transmitted with length between 1024 and maxsize (inclusive) bytes, exclusive of preamble and retried packets.

47.2.58 Transmit Unicast Packets Good Bad Register (TX_UNICAST_PACKETS_GOOD_BAD)

This register provides the number of good and bad unicast packets transmitted by the Ethernet module.

Offset 0x073C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXUCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXUCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 658. Transmit Unicast Packets Good Bad Register (TX_UNICAST_PACKETS_GOOD_BAD)

Table 721. TX_UNICAST_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TXUCASTGB	This field indicates the number of good and bad unicast packets transmitted.

47.2.59 Transmit Multicast Packets Good Bad Register (TX_MULTICAST_PACKETS_GOOD_BAD)

This register provides the number of good and bad multicast packets transmitted by the Ethernet module.

Offset 0x0740 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXMCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXMCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 659. Transmit Multicast Packets Good Bad Register (TX_MULTICAST_PACKETS_GOOD_BAD)

Table 722. TX_MULTICAST_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TXMCASTGB	This field indicates the number of good and bad multicast packets transmitted.

47.2.60 Transmit Broadcast Packets Good Bad Register (TX_BROADCAST_PACKETS_GOOD_BAD)

This register provides the number of good and bad broadcast packets transmitted by the Ethernet module.

Offset 0x0744 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXBCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXBCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 660. Transmit Broadcast Packets Good Bad Register (TX_BROADCAST_PACKETS_GOOD_BAD)

Table 723. TX_BROADCAST_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TXBCASTGB	This field indicates the number of good and bad broadcast packets transmitted

47.2.61 Transmit Underflow Error Packets Register (TX_UNDERFLOW_ERROR_PACKETS)

This register provides the number of packets aborted by the Ethernet module because of packets underflow error.

Offset 0x0748 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXUNDRFLW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXUNDRFLW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 661. Transmit Underflow Error Packets Register (TX_UNDERFLOW_ERROR_PACKETS)

Table 724. TX_UNDERFLOW_ERROR_PACKETS field descriptions

Field	Description
0:31 TXUNDRFLW	This field indicates the number of packets aborted because of packets underflow error.

47.2.62 Transmit Single Collision Good Packets Register (TX_SINGLE_COLLISION_GOOD_PACKETS)

This register provides the number of successfully transmitted packets by the Ethernet module after a single collision in the half-duplex mode.

Offset 0x074C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXSNGLCOLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXSNGLCOLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 662. Transmit Single Collision Good Packets Register (TX_SINGLE_COLLISION_GOOD_PACKETS)

Table 725. TX_SINGLE_COLLISION_GOOD_PACKETS field descriptions

Field	Description
0:31 TXSNGLCOLG	This field indicates the number of successfully transmitted packets after a single collision in the half-duplex mode.

47.2.63 Transmit Multiple Collision Good Packets Register (TX_MULTIPLE_COLLISION_GOOD_PACKETS)

This register provides the number of successfully transmitted packets by the Ethernet module after multiple collisions in the half-duplex mode.

Offset 0x0750 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXMULTLCOLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXMULTLCOLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 663. Transmit Multiple Collision Good Packets Register (TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Table 726. TX_MULTIPLE_COLLISION_GOOD_PACKETS field descriptions

Field	Description
0:31 TXMULTLCOLG	This field indicates the number of successfully transmitted packets after multiple collisions in the half-duplex mode.

47.2.64 Transmit Deferred Packets Register (TX_DEFERRED_PACKETS)

This register provides the number of successfully transmitted by the Ethernet module after a deferral in the half-duplex mode.

Offset 0x0754 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXDEFRD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDEFRD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 664. Transmit Deferred Packets Register (TX_DEFERRED_PACKETS)

Table 727. TX_DEFERRED_PACKETS field descriptions

Field	Description
0:31 TXDEFRD	This field indicates the number of successfully transmitted after a deferral in the half-duplex mode.

47.2.65 Transmit Late Collision Packets Register (TX_LATE_COLLISION_PACKETS)

This register provides the number of packets aborted by the Ethernet module because of late collision error.

Offset 0x0758 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXLATECOL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXLATECOL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 665. Transmit Late Collision Packets Register (TX_LATE_COLLISION_PACKETS)

Table 728. TX_LATE_COLLISION_PACKETS field descriptions

Field	Description
0:31 TXLATECOL	This field indicates the number of packets aborted because of late collision error.

47.2.66 Transmit Excessive Collision Packets Register (TX_EXCESSIVE_COLLISION_PACKETS)

This register provides the number of packets aborted by the Ethernet module because of excessive (16) collision errors.

Offset 0x075C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXEXSCOL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXEXSCOL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 666. Transmit Excessive Collision Packets Register (TX_EXCESSIVE_COLLISION_PACKETS)

Table 729. TX_EXCESSIVE_COLLISION_PACKETS field descriptions

Field	Description
0:31 TXEXSCOL	This field indicates the number of packets aborted because of excessive (16) collision errors.

47.2.67 Transmit Carrier Error Packets Register (TX_CARRIER_ERROR_PACKETS)

This register provides the number of packets aborted by the Ethernet module because of carrier sense error (no carrier or loss of carrier).

Offset 0x0760 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXCARR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCARR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 667. Transmit Carrier Error Packets Register (TX_CARRIER_ERROR_PACKETS)

Table 730. TX_CARRIER_ERROR_PACKETS field descriptions

Field	Description
0:31 TXCARR	This field indicates the number of packets aborted because of carrier sense error (no carrier or loss of carrier).

47.2.68 Transmit Octet Count Good Register (TX_OCTET_COUNT_GOOD)

This register provides the number of bytes transmitted by the Ethernet module, exclusive of preamble, only in good packets.

Offset 0x0764 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXOCTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXOCTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 668. Transmit Octet Count Good Register (TX_OCTET_COUNT_GOOD)

Table 731. TX_OCTET_COUNT_GOOD field descriptions

Field	Description
0:31 TXOCTG	This field indicates the number of bytes transmitted, exclusive of preamble, only in good packets.

47.2.69 Transmit Packet Count Good Register (TX_PACKET_COUNT_GOOD)

This register provides the number of good packets transmitted by the Ethernet module.

Offset 0x0768 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXPKTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXPKTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 669. Transmit Packet Count Good Register (TX_PACKET_COUNT_GOOD)

Table 732. TX_PACKET_COUNT_GOOD field descriptions

Field	Description
0:31 TXPKTG	This field indicates the number of good packets transmitted.

47.2.70 Transmit Excessive Deferral Error Register (TX_EXCESSIVE_DEFERRAL_ERROR)

This register provides the number of packets aborted by the Ethernet module because of excessive deferral error (deferred for more than two max-sized packet times).

Offset 0x076C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXEXSDEF															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXEXSDEF															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 670. Transmit Excessive Deferral Error Register (TX_EXCESSIVE_DEFERRAL_ERROR)

Table 733. TX_EXCESSIVE_DEFERRAL_ERROR field descriptions

Field	Description
0:31 TXEXSDEF	This field indicates the number of packets aborted because of excessive deferral error (deferred for more than two max-sized packet times).

47.2.71 Transmit Pause Packets Register (TX_PAUSE_PACKETS)

This register provides the number of good Pause packets transmitted by the Ethernet module.

Offset 0x0770 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXPAUSE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXPAUSE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 671. Transmit Pause Packets Register (TX_PAUSE_PACKETS)

Table 734. TX_PAUSE_PACKETS field descriptions

Field	Description
0:31 TXPAUSE	This field indicates the number of good Pause packets transmitted.

47.2.72 Transmit VLAN Packets Good Register (TX_VLAN_PACKETS_GOOD)

This register provides the number of good VLAN packets transmitted by the Ethernet module.

Offset 0x0774 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXVLANG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXVLANG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 672. Transmit VLAN Packets Good Register (TX_VLAN_PACKETS_GOOD)

Table 735. TX_VLAN_PACKETS_GOOD field descriptions

Field	Description
0:31 TXVLANG	This field provides the number of good VLAN packets transmitted.

47.2.73 Transmit OSize Packets Good Register (TX_OSIZE_PACKETS_GOOD)

This register provides the number of packets transmitted by the Ethernet module without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in S2KP bit of the MAC_Configuration register).

Offset 0x0778 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXOSIZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXOSIZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 673. Transmit OSize Packets Good Register (TX_OSIZE_PACKETS_GOOD)

Table 736. TX_OSIZE_PACKETS_GOOD field descriptions

Field	Description
0:31 TXOSIZG	This field indicates the number of packets transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in S2KP bit of the MAC_Configuration register).

47.2.74 Receive Packets Count Good Bad Register (RX_PACKETS_COUNT_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module.

Offset 0x0780 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 674. Receive Packets Count Good Bad Register (RX_PACKETS_COUNT_GOOD_BAD)

Table 737. RX_PACKETS_COUNT_GOOD_BAD field descriptions

Field	Description
0:31 RXPKTGB	This field indicates the number of good and bad packets received.

47.2.75 Receive Octet Count Good Bad Register (RX_OCTET_COUNT_GOOD_BAD)

This register provides the number of bytes received by DWC_ther_qos, exclusive of preamble, in good and bad packets.

Offset 0x0784 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 675. Receive Octet Count Good Bad Register (RX_OCTET_COUNT_GOOD_BAD)

Table 738. RX_OCTET_COUNT_GOOD_BAD field descriptions

Field	Description
0:31 RXOCTGB	This field indicates the number of bytes received, exclusive of preamble, in good and bad packets.

47.2.76 Receive Octet Count Good Register (RX_OCTET_COUNT_GOOD)

This register provides the number of bytes received by the Ethernet module, exclusive of preamble, only in good packets.

Offset 0x0788 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXOCTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXOCTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 676. Receive Octet Count Good Register (RX_OCTET_COUNT_GOOD)

Table 739. RX_OCTET_COUNT_GOOD field descriptions

Field	Description
0:31 RXOCTG	This field indicates the number of bytes received, exclusive of preamble, in good and bad packets.

47.2.77 Receive Broadcast Packets Good Register (RX_BROADCAST_PACKETS_GOOD)

This register provides the number of good broadcast packets received by the Ethernet module.

Offset 0x078C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXBCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXBCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 677. Receive Broadcast Packets Good Register (RX_BROADCAST_PACKETS_GOOD)

Table 740. RX_BROADCAST_PACKETS_GOOD field descriptions

Field	Description
0:31 RXBCASTG	This field indicates the number of good broadcast packets received.

47.2.78 Receive Multicast Packets Good Register (RX_MULTICAST_PACKETS_GOOD)

This register provides the number of good multicast packets received by the Ethernet module.

Offset 0x0790 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXMCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXMCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 678. Receive Multicast Packets Good Register (RX_MULTICAST_PACKETS_GOOD)

Table 741. RX_MULTICAST_PACKETS_GOOD field descriptions

Field	Description
0:31 RXMCASTG	This field indicates the number of good multicast packets received.

47.2.79 Receive CRC Error Packets Register (RX_CRC_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with CRC error.

Offset 0x0794 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXMCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXMCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 679. Receive CRC Error Packets Register (RX_CRC_ERROR_PACKETS)

Table 742. RX_CRC_ERROR_PACKETS field descriptions

Field	Description
0:31 RXMCASTG	This field indicates the number of good multicast packets received.

47.2.80 Receive Alignment Error Packets Register (RX_ALIGNMENT_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with alignment (dribble) error. It is valid only in 10/100 mode.

Offset 0x0798 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXALGNERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXALGNERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 680. Receive Alignment Error Packets Register (RX_ALIGNMENT_ERROR_PACKETS)

Table 743. RX_ALIGNMENT_ERROR_PACKETS field descriptions

Field	Description
0:31 RXALGNERR	This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

47.2.81 Receive Runt Error Packets Register (RX_RUNT_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with runt (length less than 64 bytes and CRC error) error.

Offset 0x079C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXRUNTERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXRUNTERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 681. Receive Runt Error Packets Register (RX_RUNT_ERROR_PACKETS)

Table 744. RX_RUNT_ERROR_PACKETS field descriptions

Field	Description
0:31 RXRUNTERR	This field indicates the number of packets received with runt (length less than 64 bytes and CRC error) error.

47.2.82 Receive Jabber Error Packets Register (RX_JABBER_ERROR_PACKETS)

This register provides the number of giant packets received by the Ethernet module with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

Offset 0x07A0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXJABERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXJABERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 682. Receive Jabber Error Packets Register (RX_JABBER_ERROR_PACKETS)

Table 745. RX_JABBER_ERROR_PACKETS field descriptions

Field	Description
0:31 RXJABERR	This field indicates the number of giant packets received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

47.2.83 Receive Undersize Packets Good Register (RX_UNDERSIZE_PACKETS_GOOD)

This register provides the number of packets received by the Ethernet module with length less than 64 bytes, without any errors.

Offset 0x07A4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUNDERSZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUNDERSZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 683. Receive Undersize Packets Good Register (RX_UNDERSIZE_PACKETS_GOOD)

Table 746. RX_UNDERSIZE_PACKETS_GOOD field descriptions

Field	Description
0:31 RXUNDERSZG	This field indicates the number of giant packets received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

47.2.84 Receive Oversize Packets Good Register (RX_OVERSIZE_PACKETS_GOOD)

This register provides the number of packets received by the Ethernet module without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in the S2KP bit of the MAC_Configuration register).

Offset 0x07A8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXOVERSZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXOVERSZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 684. Receive Oversize Packets Good Register (RX_OVERSIZE_PACKETS_GOOD)

Table 747. RX_OVERSIZE_PACKETS_GOOD field descriptions

Field	Description
0:31 RXOVERSZG	This field indicates the number of packets received without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in the S2KP bit of the MAC_Configuration register).

47.2.85 Receive 64Octets Packets Good Bad Register (RX_64OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length 64 bytes, exclusive of the preamble.

Offset 0x07AC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX64OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX64OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 685. Receive 64Octets Packets Good Bad Register (RX_64OCTETS_PACKETS_GOOD_BAD)

Table 748. RX_64OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX64OCTGB	This field indicates the number of good and bad packets received with length 64 bytes, exclusive of the preamble.

47.2.86 Receive 65To127Octets Packets Good Bad Register (RX_65TO127OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

Offset 0x07B0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX65_127OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX65_127OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 686. Receive 65To127Octets Packets Good Bad Register (RX_65TO127OCTETS_PACKETS_GOOD_BAD)

Table 749. RX_65TO127OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX65_127OCTGB	This field indicates the number of good and bad packets received with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

47.2.87 Receive 128To255Octets Packets Good Bad Register (RX_128TO255OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

Offset 0x07B4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX128_255OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX128_255OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 687. Receive 128To255Octets Packets Good Bad Register (RX_128TO255OCTETS_PACKETS_GOOD_BAD)

Table 750. RX_128TO255OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX128_255OCTGB	This field indicates the number of good and bad packets received with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

47.2.88 Receive 256To511Octets Packets Good Bad Register (RX_256TO511OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

Offset 0x07B8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX256_511OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX256_511OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 688. Receive 256To511Octets Packets Good Bad Register (RX_256TO511OCTETS_PACKETS_GOOD_BAD)

Table 751. RX_256TO511OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX256_511OCTGB	This field indicates the number of good and bad packets received with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

47.2.89 Receive 512To1023Octets Packets Good Bad Register (RX_512TO1023OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

Offset 0x07BC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX512_1023OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX512_1023OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 689. Receive 512To1023Octets Packets Good Bad Register (RX_512TO1023OCTETS_PACKETS_GOOD_BAD)

Table 752. RX_512TO1023OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX512_1023OCTGB	This field indicates the number of good and bad packets received with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

47.2.90 Receive 1024ToMax Octets Packets Good bad Register (RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

Offset 0x07C0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX1024_MAXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX1024_MAXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 690. Receive 1024ToMax Octets Packets Good bad Register (RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

Table 753. RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX1024_MAXOCTGB	This field indicates the number of good and bad packets received with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

47.2.91 Receive Unicast Packets Good Register (RX_UNICAST_PACKETS_GOOD)

This register provides the number of good unicast packets received by the Ethernet module.

Offset 0x07C4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 691. Receive Unicast Packets Good Register (RX_UNICAST_PACKETS_GOOD)

Table 754. RX_UNICAST_PACKETS_GOOD field descriptions

Field	Description
0:31 RXUCASTG	This field indicates the number of good unicast packets received.

47.2.92 Receive Length Error Packets (RX_LENGTH_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with length error (Length Type field not equal to packet size), for all packets with valid length field.

Offset 0x07C8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXLENERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXLENERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 692. Receive Length Error Packets (RX_LENGTH_ERROR_PACKETS)

Table 755. RX_LENGTH_ERROR_PACKETS field descriptions

Field	Description
0:31 RXLENERR	This field indicates the number of packets received with length error (Length Type field not equal to packet size), for all packets with valid length field.

47.2.93 Receive Out Of Range Type Packets (RX_OUT_OF_RANGE_TYPE_PACKETS)

This register provides the number of packets received by the Ethernet module with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

Offset 0x07CC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXOUTOFRNG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXOUTOFRNG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 693. Receive Out Of Range Type Packets (RX_OUT_OF_RANGE_TYPE_PACKETS)

Table 756. RX_OUT_OF_RANGE_TYPE_PACKETS field descriptions

Field	Description
0:31 RXOUTOFRNG	This field indicates the number of packets received with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

47.2.94 Receive Pause Packets Register (RX_PAUSE_PACKETS)

This register provides the number of good and valid Pause packets received by the Ethernet module.

Offset 0x07D0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXPAUSEPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXPAUSEPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 694. Receive Pause Packets Register (RX_PAUSE_PACKETS)

Table 757. RX_PAUSE_PACKETS field descriptions

Field	Description
0:31 RXPAUSEPKT	This field indicates the number of good and valid Pause packets received.

47.2.95 Receive FIFO Overflow Packets Register (RX_FIFO_OVERFLOW_PACKETS)

This register provides the number of missed received packets because of FIFO overflow in the Ethernet module.

Offset 0x07D4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXFIFOOVFL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXFIFOOVFL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 695. Receive FIFO Overflow Packets Register (RX_FIFO_OVERFLOW_PACKETS)

Table 758. RX_FIFO_OVERFLOW_PACKETS field descriptions

Field	Description
0:31 RXFIFOOVFL	This field indicates the number of missed received packets because of FIFO overflow.

47.2.96 Receive VLAN Packets Good Bad Register (RX_VLAN_PACKETS_GOOD_BAD)

This register provides the number of good and bad VLAN packets received by the Ethernet module.

Offset 0x07D8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXVLANPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXVLANPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 696. Receive VLAN Packets Good Bad Register (RX_VLAN_PACKETS_GOOD_BAD)

Table 759. RX_VLAN_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RXVLANPKTGB	This field indicates the number of good and bad VLAN packets received.

47.2.97 Receive Watchdog Error Packets Register (RX_WATCHDOG_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when JE and WD bits are reset in MAC_Configuration register), 10,240 bytes (when JE bit is set and WD bit is reset in MAC_Configuration register), 16,384 bytes (when WD bit is set in MAC_Configuration register) or the value programed in the MAC_Watchdog_Timeout register).

Offset 0x07DC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXWDGERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXWDGERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 697. Receive Watchdog Error Packets Register (RX_WATCHDOG_ERROR_PACKETS)

Table 760. RX_WATCHDOG_ERROR_PACKETS field descriptions

Field	Description
0:31 RXWDGERR	This field indicates the number of packets received with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when JE and WD bits are reset in MAC_Configuration register), 10,240 bytes (when JE bit is set and WD bit is reset in MAC_Configuration register), 16,384 bytes (when WD bit is set in MAC_Configuration register) or the value programed in the MAC_Watchdog_Timeout register).

47.2.98 Receive Error Packets Register (RX_RECEIVE_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with Receive error or Packet Extension error on the MII interface.

Offset 0x07E0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXRCVERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXRCVERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 698. Receive Error Packets Register (RX_RECEIVE_ERROR_PACKETS)

Table 761. RX_RECEIVE_ERROR_PACKETS field descriptions

Field	Description
0:31 RXRCVERR	This field indicates the number of packets received with Receive error or Packet Extension error on the MII interface.

47.2.99 Receive Control Packets Good Register (RX_CONTROL_PACKETS_GOOD)

This register provides the number of good control packets received by the Ethernet module.

Offset 0x07E4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXCTRLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXCTRLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 699. Receive Control Packets Good Register (RX_CONTROL_PACKETS_GOOD)

Table 762. RX_CONTROL_PACKETS_GOOD field descriptions

Field	Description
0:31 RXCTRLG	This field indicates the number of good control packets received.

47.2.100 Transmit LPI USEC Counter (TX_LPI_USEC_CNTR)

This register provides the number of microseconds Tx LPI is asserted by the Ethernet module.

Offset 0x07EC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXLPIUSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXLPIUSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 700. Transmit LPI USEC Counter (TX_LPI_USEC_CNTR)

Table 763. TX_LPI_USEC_CNTR field descriptions

Field	Description
0:31 TXLPIUSC	This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

47.2.101 Transmit LPI Transaction Counter Register (TX_LPI_TRAN_CNTR)

This register provides the number of times the Ethernet module has entered Tx LPI.

Offset 0x07F0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXLPITRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXLPITRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 701. Transmit LPI Transaction Counter Register (TX_LPI_TRAN_CNTR)

Table 764. TX_LPI_TRAN_CNTR descriptions

Field	Description
0:31 TXLPITRC	This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate Mode (because of LPITXA bit set in the LPI Control and Status register), the counter will increment.

47.2.102 Receive LPI USEC Counter (RX_LPI_USEC_CNTR)

This register provides the number of microseconds Rx LPI is sampled by the Ethernet module.

Offset 0x07F4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXLPIUSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXLPIUSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 702. Receive LPI USEC Counter (RX_LPI_USEC_CNTR)

Table 765. RX_LPI_USEC_CNTR field descriptions

Field	Description
0:31 RXLPIUSC	This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

47.2.103 Receive LPI Transaction Counter Register (RX_LPI_TRAN_CNTR)

This register provides the number of times the Ethernet module has entered Rx LPI.

Offset 0x07F8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXLPITRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXLPITRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 703. Receive LPI Transaction Counter Register (RX_LPI_TRAN_CNTR)

Table 766. RX_LPI_TRAN_CNTR field descriptions

Field	Description
0:31 RXLPITRC	This field indicates the number of times Rx LPI Entry has occurred.

47.2.104 MMC IPC Receive Interrupt Mask Register (MMC_IPC_RX_INTERRUPT_MASK)

This register maintains the mask for the interrupt generated from the receive IPC statistic counters.

Offset 0x0800 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXICMPEROIM	RXICMPGOIM	RXTCPEROIM	RXTCPGOIM	RXUDPEROIM	RXUDPGOIM	RXIPV6NOPAYOIM	RXIPV6HEROIM	RXIPV6GOIM	RXIPV4UDSBLOIM	RXIPV4FRAGOIM	RXIPV4NOPAYOIM	RXIPV4HEROIM	RXIPV4GOIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	RXICMPERPIM	RXICMPGPIM	RXTCPERPIM	RXTCPGPIM	RXUDPERPIM	RXUDPGPIM	RXIPV6NOPAYPIM	RXIPV6HERPIM	RXIPV6GPIM	RXIPV4UDSBLPIM	RXIPV4FRAGPIM	RXIPV4NOPAYPIM	RXIPV4HERPIM	RXIPV4GPIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 704. MMC IPC Receive Interrupt Mask Register (MMC_IPC_RX_INTERRUPT_MASK)

Table 767. MMC_IPC_RX_INTERRUPT_MASK field descriptions

Field	Description
2 RXICMPEROIM	MMC Receive ICMP Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value.
3 RXICMPGOIM	MMC Receive ICMP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value.
4 RXTCPEROIM	MMC Receive TCP Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value.
5 RXTCPGOIM	MMC Receive TCP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxtcp_gd_octets counter reaches half of the maximum value or the maximum value.
6 RXUDPEROIM	MMC Receive UDP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxudp_err_octets counter reaches half of the maximum value or the maximum value.
7 RXUDPGOIM	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value.
8 RXIPV6NOPAYOIM	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value.

Table 767. MMC_IPC_RX_INTERRUPT_MASK field descriptions (continued)

Field	Description
9 RXIPV6HEROIM	MMC Receive IPV6 Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value.
10 RXIPV6GOIM	MMC Receive IPV6 Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value.
11 RXIPV4UDSBLOIM	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value.
12 RXIPV4FRAGOIM	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value.
13 RXIPV4NOPAYOIM	MMC Receive IPV4 No Payload Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value.
14 RXIPV4HEROIM	MMC Receive IPV4 Header Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value.
15 RXIPV4GOIM	MMC Receive IPV4 Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value.
18 RXICMPERPIM	MMC Receive ICMP Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxicmp_err_pkts counter reaches half of the maximum value or the maximum value.
19 RXICMPGPIM	MMC Receive ICMP Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxicmp_gd_pkts counter reaches half of the maximum value or the maximum value.
20 RXTCPERPIM	MMC Receive TCP Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxtcp_err_pkts counter reaches half of the maximum value or the maximum value.
21 RXTCPGPIM	MMC Receive TCP Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxtcp_gd_pkts counter reaches half of the maximum value or the maximum value.
22 RXUDPERPIM	MMC Receive UDP Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxudp_err_pkts counter reaches half of the maximum value or the maximum value.
23 RXUDPGPIM	MMC Receive UDP Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxudp_gd_pkts counter reaches half of the maximum value or the maximum value.
24 RXIPV6NOPAYPIM	MMC Receive IPV6 No Payload Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_nopay_pkts counter reaches half of the maximum value or the maximum value.

Table 767. MMC_IPC_RX_INTERRUPT_MASK field descriptions (continued)

Field	Description
25 RXIPV6HERPIM	MMC Receive IPV6 Header Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_hdrerr_pkts counter reaches half of the maximum value or the maximum value.
26 RXIPV6GPIM	MMC Receive IPV6 Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_gd_pkts counter reaches half of the maximum value or the maximum value.
27 RXIPV4UDSBLPIM	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_udsbl_pkts counter reaches half of the maximum value or the maximum value.
28 RXIPV4FRAGPIM	MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_frag_pkts counter reaches half of the maximum value or the maximum value.
29 RXIPV4NOPAYPIM	MMC Receive IPV4 No Payload Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_nopay_pkts counter reaches half of the maximum value or the maximum value.
30 RXIPV4HERPIM	MMC Receive IPV4 Header Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_hdrerr_pkts counter reaches half of the maximum value or the maximum value.
31 RXIPV4GPIM	MMC Receive IPV4 Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_gd_pkts counter reaches half of the maximum value or the maximum value.

47.2.105 MMC IPC Receive Interrupt Register (MMC_IPC_RX_INTERRUPT)

This register maintains the interrupt that the receive IPC statistic counters generate.

Offset 0x0808 Access: User Read

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXICMPEROIS	RXICMPGOIS	RXTCPEROIS	RXTCPGOIS	RXUDPEROIS	RXUDPGOIS	RXIPV6NOPAYOIS	RXIPV6HEROIS	RXIPV6GOIS	RXIPV4UDSBLOIS	RXIPV4FRAGOIS	RXIPV4NOPAYOIS	RXIPV4HEROIS	RXIPV4GOIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	RXICMPERPIS	RXICMPGPIS	RXTCPERPIS	RXTCPGPIS	RXUDPERPIS	RXUDPGPIS	RXIPV6NOPAYPIS	RXIPV6HERPIS	RXIPV6GPIS	RXIPV4UDSBLPIS	RXIPV4FRAGPIS	RXIPV4NOPAYPIS	RXIPV4HERPIS	RXIPV4GPIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 705. MMC IPC Receive Interrupt Register (MMC_IPC_RX_INTERRUPT)

Table 768. MMC_IPC_RX_INTERRUPT field descriptions

Name	Description
2 RXICMPEROIS	MMC Receive ICMP Error Octet Counter Interrupt Status This bit is set when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
3 RXICMPGOIS	MMC Receive ICMP Good Octet Counter Interrupt Status This bit is set when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
4 RXTCPEROIS	MMC Receive TCP Error Octet Counter Interrupt Status This bit is set when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5 RXTCPGOIS	MMC Receive TCP Good Octet Counter Interrupt Status This bit is set when the rxtcp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
6 RXUDPEROIS	MMC Receive UDP Error Octet Counter Interrupt Status This bit is set when the rxudp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 768. MMC_IPC_RX_INTERRUPT field descriptions (continued)

Name	Description
7 RXUDPGOIS	MMC Receive UDP Good Octet Counter Interrupt Status This bit is set when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
8 RXIPV6NOPAYOIS	MMC Receive IPV6 No Payload Octet Counter Interrupt Status This bit is set when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
9 RXIPV6HEROIS	MMC Receive IPV6 Header Error Octet Counter Interrupt Status This bit is set when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
10 RXIPV6GOIS	MMC Receive IPV6 Good Octet Counter Interrupt Status This bit is set when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
11 RXIPV4UDSBLOIS	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status This bit is set when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
12 RXIPV4FRAGOIS	MMC Receive IPV4 Fragmented Octet Counter Interrupt Status This bit is set when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
13 RXIPV4NOPAYOIS	MMC Receive IPV4 No Payload Octet Counter Interrupt Status This bit is set when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
14 RXIPV4HEROIS	MMC Receive IPV4 Header Error Octet Counter Interrupt Status This bit is set when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
15 RXIPV4GOIS	MMC Receive IPV4 Good Octet Counter Interrupt Status This bit is set when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
18 RXICMPERPIS	MMC Receive ICMP Error Packet Counter Interrupt Status This bit is set when the rxicmp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 768. MMC_IPC_RX_INTERRUPT field descriptions (continued)

Name	Description
19 RXICMPGPIS	MMC Receive ICMP Good Packet Counter Interrupt Status This bit is set when the rxicmp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
20 RXTCPERPIS	MMC Receive TCP Error Packet Counter Interrupt Status This bit is set when the rxtcp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21 RXTCPGPIS	MMC Receive TCP Good Packet Counter Interrupt Status This bit is set when the rxtcp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
22 RXUDPERPIS	MMC Receive UDP Error Packet Counter Interrupt Status This bit is set when the rxudp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
23 RXUDPGPIS	MC Receive UDP Good Packet Counter Interrupt Status This bit is set when the rxudp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
24 RXIPV6NOPAYPIS	MMC Receive IPV6 No Payload Packet Counter Interrupt Status This bit is set when the rxipv6_nopay_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
25 RXIPV6HERPIS	MMC Receive IPV6 Header Error Packet Counter Interrupt Status This bit is set when the rxipv6_hdrerr_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 RXIPV6GPIS	MMC Receive IPV6 Good Packet Counter Interrupt Status This bit is set when the rxipv6_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
27 RXIPV4UDSBLPIS	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status This bit is set when the rxipv4_udsbl_pkts counter reaches half of the maximum value or the maximum value.
28 RXIPV4FRAGPIS	MMC Receive IPV4 Fragmented Packet Counter Interrupt Status This bit is set when the rxipv4_frag_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 RXIPV4NOPAYPIS	MMC Receive IPV4 No Payload Packet Counter Interrupt Status This bit is set when the rxipv4_nopay_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 768. MMC_IPC_RX_INTERRUPT field descriptions (continued)

Name	Description
30 RXIPV4HERPIS	MMC Receive IPV4 Header Error Packet Counter Interrupt Status This bit is set when the rxipv4_hdrerr_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 RXIPV4GPIS	MMC Receive IPV4 Good Packet Counter Interrupt Status This bit is set when the rxipv4_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.106 Receive IPv4 Good Packets Register (RXIPV4_GOOD_PACKETS)

This register provides the number of good IPv4 datagrams received by the Ethernet module with the TCP, UDP, or ICMP payload.

Offset 0x0810 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4GDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4GDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 706. Receive IPv4 Good Packets Register (RXIPV4_GOOD_PACKETS)

Table 769. RXIPV4_GOOD_PACKETS field descriptions

Field	Description
0:31 RXIPV4GDPKT	This field indicates the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

47.2.107 Receive IPv4 Header Error Packets (RXIPV4_HEADER_ERROR_PACKETS)

This register provides the number of IPv4 datagrams received by the Ethernet module with header (checksum, length, or version mismatch) errors.

Offset 0x0814 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4HDRERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4HDRERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 707. Receive IPv4 Header Error Packets (RXIPV4_HEADER_ERROR_PACKETS)

Table 770. RXIPV4_HEADER_ERROR_PACKETS field descriptions

Field	Description
0:31 RXIPV4HDRERRPKT	This field indicates the number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors.

47.2.108 Receive IPv4 No Payload Packets Register (RXIPV4_NO_PAYLOAD_PACKETS)

This register provides the number of IPv4 datagram packets received by the Ethernet module that did not have a TCP, UDP, or ICMP payload.

Offset 0x0818 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4NOPAYPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4NOPAYPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 708. Receive IPv4 No Payload Packets Register (RXIPV4_NO_PAYLOAD_PACKETS)

Table 771. RXIPV4_NO_PAYLOAD_PACKETS field descriptions

Field	Description
0:31 RXIPV4NOPAYPKT	This field indicates the number of IPv4 datagram packets received that did not have a TCP, UDP, or ICMP payload.

47.2.109 Receive IPv4 Fragmented Packets Register (RXIPV4_FRAGMENTED_PACKETS)

This register provides the number of good IPv4 datagrams received by the Ethernet module with fragmentation.

Offset 0x081C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4FRAGPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4FRAGPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 709. Receive IPv4 Fragmented Packets Register (RXIPV4_FRAGMENTED_PACKETS)

Table 772. RXIPV4_FRAGMENTED_PACKETS field descriptions

Field	Description
0:31 RXIPV4FRAGPKT	This field indicates the number of good IPv4 datagrams received with fragmentation.

47.2.110 Receive IPv4 UDP Checksum Disabled Packets Register (RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS)

This register provides the number of good IPv4 datagrams received by the Ethernet module that had a UDP payload with checksum disabled.

Offset 0x0820 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4UDSBLPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4UDSBLPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 710. Receive IPv4 UDP Checksum Disabled Packets Register (RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS)

Table 773. RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS field descriptions

Field	Description
0:31 RXIPV4UDSBLPKT	This field indicates the number of good IPv4 datagrams received that had a UDP payload with checksum disabled.

47.2.111 Receive IPv6 Good Packets Register (RXIPV6_GOOD_PACKETS)

This register provides the number of good IPv6 datagrams received by the Ethernet module with the TCP, UDP, or ICMP payload.

Offset 0x0824 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6GDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6GDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 711. Receive IPv6 Good Packets Register (RXIPV6_GOOD_PACKETS)

Table 774. RXIPV6_GOOD_PACKETS field descriptions

Field	Description
0:31 RXIPV6GDPKT	This field indicates the number of good IPv6 datagrams received with the TCP, UDP, or ICMP payload.

47.2.112 Receive IPv6 Header Error Packets (RXIPV6_HEADER_ERROR_PACKETS)

This register provides the number of IPv6 datagrams received by the Ethernet module with header (length or version mismatch) errors.

Offset 0x0828 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6HDRERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6HDRERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 712. Receive IPv6 Header Error Packets (RXIPV6_HEADER_ERROR_PACKETS)

Table 775. RXIPV6_HEADER_ERROR_PACKETS field descriptions

Field	Description
0:31 RXIPV6HDRERRPKT	This field indicates the number of IPv6 datagrams received with header (length or version mismatch) errors.

47.2.113 Receive IPv6 Payload Packets Register (RXIPV6_NO_PAYLOAD_PACKETS)

This register provides the number of IPv6 datagram packets received by the Ethernet module that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

Offset 0x082C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6NOPAYPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6NOPAYPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 713. Receive IPv6 Payload Packets Register (RXIPV6_NO_PAYLOAD_PACKETS)

Table 776. RXIPV6_NO_PAYLOAD_PACKETS field descriptions

Field	Description
0:31 RXIPV6NOPAYPKT	This field indicates the number of IPv6 datagram packets received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

47.2.114 Receive UDP Good Packets Register (RXUDP_GOOD_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module with a good UDP payload. This counter is not updated when the RxIPv4_UDP_Checksum_Disabled_Packets counter is incremented.

Offset 0x0830 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUDPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUDPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 714. Receive UDP Good Packets Register (RXUDP_GOOD_PACKETS)

Table 777. RXUDP_GOOD_PACKETS field descriptions

Field	Description
0:31 RXUDPGDPKT	This field indicates the number of good IP datagrams received with a good UDP payload. This counter is not updated when the RxIPv4_UDP_Checksum_Disabled_Packets counter is incremented.

47.2.115 Receive UDP Error Packets Register (RXUDP_ERROR_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module whose UDP payload has a checksum error.

Offset 0x0834 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUDPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUDPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 715. Receive UDP Error Packets Register (RXUDP_ERROR_PACKETS)

Table 778. RXUDP_ERROR_PACKETS field descriptions

Field	Description
0:31 RXUDPERRPKT	This field indicates the number of good IP datagrams received whose UDP payload has a checksum error.

47.2.116 Receive TCP Good Packets Register (RXTCP_GOOD_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module with a good TCP payload.

Offset 0x0838 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXTCPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXTCPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 716. Receive TCP Good Packets Register (RXTCP_GOOD_PACKETS)

Table 779. RXTCP_GOOD_PACKETS field descriptions

Field	Description
0:31 RXTCPGDPKT	This field indicates the number of good IP datagrams received with a good TCP payload.

47.2.117 Receive TCP Error Packets Register (RXTCP_ERROR_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module whose TCP payload has a checksum error.

Offset 0x083C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXTCPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXTCPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 717. Receive TCP Error Packets Register (RXTCP_ERROR_PACKETS)

Table 780. RXTCP_ERROR_PACKETS field descriptions

Field	Description
0:31 RXTCPERRPKT	This field indicates the number of good IP datagrams received whose TCP payload has a checksum error.

47.2.118 Receive ICMP Good Packets Register (RXICMP_GOOD_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module with a good ICMP payload.

Offset 0x0840 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXICMPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXICMPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 718. Receive ICMP Good Packets Register (RXICMP_GOOD_PACKETS)

Table 781. RXICMP_GOOD_PACKETS field descriptions

Field	Description
0:31 RXICMPGDPKT	This field indicates the number of good IP datagrams received with a good ICMP payload

47.2.119 Receive ICMP Error Packets Register (RXICMP_ERROR_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module whose ICMP payload has a checksum error.

Offset 0x0844 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXICMPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXICMPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 719. Receive ICMP Error Packets Register (RXICMP_ERROR_PACKETS)

Table 782. RXICMP_ERROR_PACKETS field descriptions

Field	Description
0:31 RXICMPERRPKT	This field indicates the number of good IP datagrams received whose ICMP payload has a checksum error.

47.2.120 Receive IPv4 Good Octets Register (RXIPV4_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x0850 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4GDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4GDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 720. Receive IPv4 Good Octets Register (RXIPV4_GOOD_OCTETS)

Table 783. RXIPV4_GOOD_OCTETS field descriptions

Field	Description
0:31 RXIPV4GDOCT	This field indicates the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

47.2.121 Receive IPv4 Header Error Octets Register (RXIPV4_HEADER_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x0854 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4HDRERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4HDRERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 721. Receive IPv4 Header Error Octets Register (RXIPV4_HEADER_ERROR_OCTETS)

Table 784. RXIPV4_HEADER_ERROR_OCTETS field descriptions

Field	Description
0:31 RXIPV4HDRERROCT	This field indicates the number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

47.2.122 Receive IPv4 No Payload Octets Register (RXIPV4_NO_PAYLOAD_OCTETS)

This register provides the number of bytes received by the Ethernet module in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x0858 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4NOPAYOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4NOPAYOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 722. Receive IPv4 No Payload Octets Register (RXIPV4_NO_PAYLOAD_OCTETS)

Table 785. RXIPV4_NO_PAYLOAD_OCTETS field descriptions

Field	Description
0:31 RXIPV4NOPAYOCT	This field indicates the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

47.2.123 Receive IPv4 Fragmented Octets Register (RXIPV4_FRAGMENTED_OCTETS)

This register provides the number of bytes received by the Ethernet module in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x085C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4FRAGOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4FRAGOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 723. Receive IPv4 Fragmented Octets Register (RXIPV4_FRAGMENTED_OCTETS)

Table 786. RXIPV4_FRAGMENTED_OCTETS field descriptions

Field	Description
0:31 RXIPV4FRAGOCT	This field indicates the number of bytes received in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

47.2.124 Receive IPv4 UDP Checksum Disable Octets Register (RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS)

This register provides the number of bytes received by the Ethernet module in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x0860 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4UDSBLOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4UDSBLOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 724. Receive IPv4 UDP Checksum Disable Octets Register (RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS)

Table 787. RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS field descriptions

Field	Description
0:31 RXIPV4UDSBLOCT	This field indicates the number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

47.2.125 Receive IPv6 Good Octets Register (RXIPV6_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x0864 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6GDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6GDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 725. Receive IPv6 Good Octets Register (RXIPV6_GOOD_OCTETS)

Table 788. RXIPV6_GOOD_OCTETS field descriptions

Field	Description
0:31 RXIPV6GDOCT	This field indicates the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

47.2.126 Receive Header Error Octets Register (RXIPV6_HEADER_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x0868 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6HDRERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6HDRERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 726. Receive Header Error Octets Register (RXIPV6_HEADER_ERROR_OCTETS)

Table 789. RXIPV6_HEADER_ERROR_OCTETS field descriptions

Field	Description
0:31 RXIPV6HDRERROCT	This field indicates the number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

47.2.127 Receive IPv6 No Payload Octets Register (RXIPV6_NO_PAYLOAD_OCTETS)

This register provides the number of bytes received by the Ethernet module in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x086C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6NOPAYOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6NOPAYOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 727. Receive IPv6 No Payload Octets Register (RXIPV6_NO_PAYLOAD_OCTETS)

Table 790. RXIPV6_NO_PAYLOAD_OCTETS field descriptions

Field	Description
0:31 RXIPV6NOPAYOCT	This field indicates the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

47.2.128 Receive UDP Good Octets Register (RXUDP_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in a good UDP segment. This counter does not count IP header bytes.

Offset 0x0870 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUDPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUDPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 728. Receive UDP Good Octets Register (RXUDP_GOOD_OCTETS)

Table 791. RXUDP_GOOD_OCTETS field descriptions

Field	Description
0:31 RXUDPGDOCT	This field indicates the number of bytes received in a good UDP segment. This counter does not count IP header bytes.

47.2.129 Receive UDP Error Octets Register (RXUDP_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in a UDP segment that had checksum errors. This counter does not count IP header bytes.

Offset 0x0874 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUDPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUDPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 729. Receive UDP Error Octets Register (RXUDP_ERROR_OCTETS)

Table 792. RXUDP_ERROR_OCTETS field descriptions

Field	Description
0:31 RXUDPERROCT	This field indicates the number of bytes received in a UDP segment that had checksum errors. This counter does not count IP header bytes.

47.2.130 Receive TCP Good Octets Register (RXTCP_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in a good TCP segment. This counter does not count IP header bytes.

Offset 0x0878 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXTCPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXTCPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 730. Receive TCP Good Octets Register (RXTCP_GOOD_OCTETS)

Table 793. RXTCP_GOOD_OCTETS field descriptions

Field	Description
0:31 RXTCPGDOCT	This field indicates the number of bytes received in a good TCP segment. This counter does not count IP header bytes.

47.2.131 Receive TCP Error Octets Register (RXTCP_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in a TCP segment that had checksum errors. This counter does not count IP header bytes.

Offset 0x087C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXTCPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXTCPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 731. Receive TCP Error Octets Register (RXTCP_ERROR_OCTETS)

Table 794. RXTCP_ERROR_OCTETS field descriptions

Field	Description
0:31 RXTCPERROCT	This field indicates the number of bytes received in a TCP segment that had checksum errors. This counter does not count IP header bytes.

47.2.132 Receive ICMP Good Octets Register (RXICMP_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in a good ICMP segment. This counter does not count IP header bytes.

Offset 0x0880 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXICMPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXICMPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 732. Receive ICMP Good Octets Register (RXICMP_GOOD_OCTETS)

Table 795. RXICMP_GOOD_OCTETS field descriptions

Field	Description
0:31 RXICMPGDOCT	This field indicates the number of bytes received in a good ICMP segment. This counter does not count IP header bytes.

47.2.133 Receive ICMP Error Octets Register (RXICMP_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

Offset 0x0884 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXICMPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXICMPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 733. Receive ICMP Error Octets Register (RXICMP_ERROR_OCTETS)

Table 796. RXICMP_ERROR_OCTETS field descriptions

Field	Description
0:31 RXICMPERROCT	This field indicates the number of bytes received in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

47.2.134 Layer 3 and Layer 4 Control 0 Register (MAC_L3_L4_CONTROL0)

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

Offset 0x0900 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	DMCHEN0	0	0	0	DMCHN0	0	0	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	0	L4PEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	L3HDBM0					L3HSBM0					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	0	L3PEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 734. Layer 3 and Layer 4 Control 0 Register (MAC_L3_L4_CONTROL0)

Table 797. MAC_L3_L4_CONTROL0 field descriptions

Field	Description
3 DMCHEN0	DMA Channel Select Enable When set, this bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When this bit is reset, the DMA channel is not decided by this filter.
7 DMCHN0	DMA Channel Number When DMCHEN is set high, this field selects the DMA Channel number to which the packet passed by this filter is routed. 0 Packet is passed by this filter to DMA CH 0 1 Packet is passed by this filter to DMA CH 1
10 L4DPIM0	Layer 4 Destination Port Inverse Match Enable When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching. This bit is valid and applicable only when the L4DPM0 bit is set high.
11 L4DPM0	Layer 4 Destination Port Match Enable When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.
12 L4SPIM0	Layer 4 Source Port Inverse Match Enable When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching. This bit is valid and applicable only when the L4SPM0 bit is set high.
13 L4SPM0	Layer 4 Source Port Match Enable When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.
15 L4PEN0	Layer 4 Protocol Enable When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching. The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Table 797. MAC_L3_L4_CONTROL0 field descriptions (continued)

Field	Description
16:20 L3HDBM0	Layer 3 IP DA Higher Bits Match **IPv4 Packets:** This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 packets. The following list describes the values of this field: 0x00 No bits are masked. 0x01 1 LSb is masked 0x02 Two LSbs are masked ... 0x1F All bits except MSb are masked. **IPv6 Packets:** Bits[29:30] of this field correspond to Bits[25:26] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM0[30:31] and L3HSBM0 bits: - 0x00 No bits are masked. - 0x01 1 LSb is masked. - 0x02 Two LSbs are masked - … - 0x7F All bits except MSb are masked. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.
21:25 L3HSBM0	Layer 3 IP SA Higher Bits Match **IPv4 Packets:** This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field: 0x00 No bits are masked. 0x01 1 LSb is masked 0x02 Two LSbs are masked … 0x1F All bits except MSb are masked. **IPv6 Packets:** This field contains Bits[27:31] of L3HSBM0. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.
26 L3DAIM0	Layer 3 IP DA Inverse Match Enable When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching. This bit is valid and applicable only when the L3DAM0 bit is set high.
27 L3DAM0	Layer 3 IP DA Match Enable When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching. **Note:** When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Table 797. MAC_L3_L4_CONTROL0 field descriptions (continued)

Field	Description
28 L3SAIM0	Layer 3 IP SA Inverse Match Enable When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching. This bit is valid and applicable only when the L3SAM0 bit is set.
29 L3SAM0	Layer 3 IP SA Match Enable When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching. **Note:** When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.
31 L3PEN0	Layer 3 Protocol Enable When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets. The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set. Value After Reset: 0x0

47.2.135 Layer 4 Address 0 Register (MAC_LAYER4_ADDRESS0)

This register holds the TCP source and destination port numbers against which a match is performed.

Offset 0x0904 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	L4DP0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L4SP0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 735. Layer4 Address 0 Register (MAC_LAYER4_ADDRESS0)

Table 798. MAC_LAYER4_ADDRESS0 field descriptions

Field	Description
0:15 L4DP0	Layer 4 Destination Port Number Field When the L4PEN0 bit is reset and the L4DPM0 bit is set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets. When the L4PEN0 and L4DPM0 bits are set in MAC_L3_L4_Control0 register, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.
16:31 L4SP0	Layer 4 Source Port Number Field When the L4PEN0 bit is reset and the L4DPM0 bit is set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets. When the L4PEN0 and L4DPM0 bits are set in MAC_L3_L4_Control0 register, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

47.2.136 Layer 3 Address 0 Register 0 Register (MAC_LAYER3_ADDRESS0_REG0)

For IPv4 packets, the Layer 3 Address 0 Register 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[96:127] of the 128-bit IP Source Address or Destination Address field.

Offset 0x0910 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	L3A00															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L3A00															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 736. Layer 3 Address 0 Register 0 Register (MAC_LAYER3_ADDRESS0_REG0)

Table 799. MAC_LAYER3_ADDRESS0_REG0 field descriptions

Field	Description
0:31 L3A00	Layer 3 Address 0 Field When the L3PEN0 and L3SAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[96:127] of the IP Source Address field in the IPv6 packets. When the L3PEN0 and L3DAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[96:127] of the IP Destination Address field in the IPv6 packets. When the L3PEN0 bit is reset and the L3SAM0 bit is set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

47.2.137 Layer3 Address1 Register 0 Register (MAC_LAYER3_ADDRESS1_REG0)

For IPv4 packets, the Layer 3 Address 1 Register 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[64:95] of the 128-bit IP Source Address or Destination Address field.

Offset 0x0914 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	L3A10															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	L3A10															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 737. Layer3 Address1 Register 0 Register (MAC_LAYER3_ADDRESS1_REG0)

Table 800. MAC_LAYER3_ADDRESS1_REG0 field descriptions

Field	Description
0:31 L3A10	Layer 3 Address 1 Field When the L3PEN0 and L3SAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[64:95] of the IP Source Address field in the IPv6 packets. When the L3PEN0 and L3DAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[64:95] of the IP Destination Address field in the IPv6 packets. When the L3PEN0 bit is reset and the L3SAM0 bit is set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

47.2.138 Layer3 Address2 Register 0 Register (MAC_LAYER3_ADDRESS2_REG0)

The Layer 3 Address 2 Register 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[32:63] of 128-bit IP Source Address or Destination Address field.

Offset 0x0918 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	L3A20															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L3A20															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 738. Layer3 Address2 Register 0 Register (MAC_LAYER3_ADDRESS2_REG0)

Table 801. MAC_LAYER3_ADDRESS2_REG0 field descriptions

Field	Description
0:31 L3A20	Layer 3 Address 2 Field When the L3PEN0 and L3SAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[32:63] of the IP Source Address field in the IPv6 packets. When the L3PEN0 and L3DAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[32:63] of the IP Destination Address field in the IPv6 packets. When the L3PEN0 bit is reset in the MAC_L3_L4_Control0 register, this field is not used.

47.2.139 Layer3 Address3 Register 0 Register (MAC_LAYER3_ADDRESS3_REG0)

The Layer 3 Address 3 Register 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[0:31] of 128-bit IP Source Address or Destination Address field.

Offset 0x091C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	L3A30															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	L3A30															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 739. Layer3 Address3 Register 0 Register (MAC_LAYER3_ADDRESS3_REG0)

Table 802. MAC_LAYER3_ADDRESS3_REG0 field descriptions

Field	Description
0:31 L3A30	Layer 3 Address 3 Field When the L3PEN0 and L3SAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[0:31] of the IP Source Address field in the IPv6 packets. When the L3PEN0 and L3DAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[0:31] of the IP Destination Address field in the IPv6 packets. When the L3PEN0 bit is reset in the MAC_L3_L4_Control0 register, this field is not used.

47.2.140 Timestamp Control Register (MAC_TIMESTAMP_CONTROL)

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the Receiver.

Offset 0x0B00 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	AV8021ASMEN	0	0	0	TXTSSTSM	0	0	0	0	0	TSENMACADDR	SNAPTYPSEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	0	0	TSADDREG	TSTRIG	TSUPDT	TSINIT	TSCFUPDT	TSENA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 740. Timestamp Control Register (MAC_TIMESTAMP_CONTROL)

Table 803. MAC_TIMESTAMP_CONTROL field descriptions

Field	Description
3 AV8021ASMEN	AV 802.1AS Mode Enable When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS mode of operation.
7 TXTSSTSM	Transmit Timestamp Status Mode When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSTSMIS bit of the MAC_TxTimestamp_Status_Nanoseconds register. When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSTSHI bit of the MAC_TxTimestamp_Status_Seconds register.
13 TSENMACADDR	Enable MAC Address for PTP Packet Filtering When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet. When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet. MAC address registers 0 to 31 is considered for unicast destination address matching.
14:15 SNAPTYPSEL	Select PTP packets for Taking Snapshots These bits, along with Bits 16 and 17, decide the set of PTP packet types for which snapshot needs to be taken. The encoding is given in Timestamp Snapshot Dependency on Register Bits Table.

Table 803. MAC_TIMESTAMP_CONTROL field descriptions (continued)

Field	Description
16 TSMSTRENA	Enable Snapshot for Messages Relevant to Master When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.
17 TSEVNTENA	Enable Timestamp Snapshot for Event Messages When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, refer to Timestamp Snapshot Dependency on Register Bits Table.
18 TSIPV4ENA	Enable Processing of PTP Packets Sent over IPv4-UDP When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.
19 TSIPV6ENA	Enable Processing of PTP Packets Sent over IPv6-UDP When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.
20 TSIPENA	Enable Processing of PTP over Ethernet Packets When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.
21 TSVER2ENA	Enable PTP Packet Processing for Version 2 Format When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.
22 TSCTRLSSR	Timestamp Digital or Binary Rollover Control When this bit is set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment must be programed correctly depending on the PTP reference clock frequency and the value of this bit.
23 TSENALL	Enable Timestamp for All Packets When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.
26 TSADDREG	Update Addend Register When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
27 TSTRIG	Enable Timestamp Interrupt Trigger When this bit is set, the timestamp interrupt is generated when the System Time becomes greater than the value written in the Target Time register. This bit is reset after the Timestamp Trigger Interrupt is generated. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

Table 803. MAC_TIMESTAMP_CONTROL field descriptions (continued)

Field	Description
28 TSUPDT	Update Timestamp When this bit is set, the system time is updated (added or subtracted) with the value specified in MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update. This bit should be zero before updating it. This bit is reset when the update is complete in hardware. The Timestamp Higher Word register (if enabled during core configuration) is not updated. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
29 TSINIT	Initialize Timestamp When this bit is set, the system time is initialized (overwritten) with the value specified in the MAC Register 80 (System Time Seconds Update Register) and MAC Register 81 (System Time Nanoseconds Update Register). This bit should be zero before it is updated. This bit is reset when the initialization is complete. The Timestamp Higher Word register (if enabled during core configuration) can only be initialized. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
30 TSCFUPDT	Fine or Coarse Timestamp Update When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.
31 TSENA	Enable Timestamp When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode. On the Receive side, the MAC processes the 1588 packets only if this bit is set.

47.2.141 Sub Second Increment Register (MAC_SUB_SECOND_INCREMENT)

In Coarse Update mode [Bit 30 in MAC_Timestamp_Control], the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

Offset 0x0B04 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SSINC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SNSINC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 741. Sub Second Increment Register (MAC_SUB_SECOND_INCREMENT)

Table 804. MAC_SUB_SECOND_INCREMENT field descriptions

Name	Description
8:15 SSINC	Sub-second Increment Value The value programed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the sub-second register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [Bit 22 (TSCTRLSSR) is set in MAC_Timestamp_Control]. When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.
16:23 SNSINC	Sub-nanosecond Increment Value This field contains the sub-nanosecond increment value, represented in nanoseconds multiplied by 2 ^ 8. This value is accumulated with the sub-nanoseconds field of the subsecond register. For example, when TSCTRLSSR field in the MAC_Timestamp_Control register is set, and if the required increment is 5.3 ns, then SSINC should be 0x05 and SNSINC should be 0x4C. This field is reserved and read-only when Enable One-Step Timestamp Feature and Enable IEEE 1588 Sub-Nanoseconds Timestamp Support features are not selected.

47.2.142 System Time Seconds Register (MAC_SYSTEM_TIME_SECONDS)

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

Offset 0x0B08 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 742. System Time Seconds Register (MAC_SYSTEM_TIME_SECONDS)

Table 805. MAC_SYSTEM_TIME_SECONDS field descriptions

Name	Description
0:31 TSS	Timestamp Second The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

47.2.143 System Time Nanoseconds Register (MAC_SYSTEM_TIME_NANOSECONDS)

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

Offset 0x0B0C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSSS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSSS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 743. System Time Nanoseconds Register (MAC_SYSTEM_TIME_NANOSECONDS)

Table 806. MAC_SYSTEM_TIME_NANOSECONDS field descriptions

Name	Description
0:31 TSSS	Timestamp Sub Seconds The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When Bit 22 is set in MAC_Timestamp_Control, each bit represents 1 ns. The maximum value is 0x3B9A_C9FF after which it rolls-over to zero.

47.2.144 System Time Seconds Update Register (MAC_SYSTEM_TIME_SECONDS_UPDATE)

The System Time Seconds Update register, along with the System Time Nanoseconds Update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in MAC_Timestamp_Control.

Offset 0x0B10 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TSS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TSS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 744. System Time Seconds Update Register (MAC_SYSTEM_TIME_SECONDS_UPDATE)

Table 807. MAC_SYSTEM_TIME_SECONDS_UPDATE field descriptions

Field	Description
0:31 TSS	Timestamp Seconds The value in this field is the sub-second part of the update. When ADDSUB is reset, this field must be programed with the subsecond part of the update value, with an accuracy based on the TSCTRLSSR bit of the MAC_Timestamp_Control register. When ADDSUB is set, then this field must be programed with the complement of the sub-second part of the update value as described below. When TSCTRLSSR is set, then the programed value must be 10^9 - <sub-second value>. When TSCTRLSSR is reset, then the programed value must be 2^31 - <sub-second_value> For example, when TSCTRLSSR bit is set and if 2.000000001 seconds need to be subtracted from the system time, then the TSS field in the MAC_Timestamp Seconds update register must be 0xFFFF_FFFE (that is, 2^32 - 2), ADDSUB bit in this register should be set, and the TSSS field must be 0x3B9A_C9FF (that is, 10^9 - 1).

47.2.145 System Time Nanoseconds Update Register (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE)

Offset 0x0B14 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ADDSUB	TSS														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TSS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 745. System Time Nanoseconds Update Register (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE)

Table 808. MAC_SYSTEM_TIME_NANOSECONDS_UPDATE field descriptions

Field	Description
0 ADDSUB	Add or Subtract Time When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.
1:31 TSS	Timestamp Sub Seconds The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When the TSCTRLSSR bit is set in the MAC_Timestamp_Control register, each bit represents 1 ns and the programed value should not exceed 0x3B9A_C9FF.

47.2.146 Timestamp Addend Register (MAC_TIMESTAMP_ADDEND)

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the MAC_Timestamp_Control register). The content of this register is added to a 32-bit accumulator in every clock cycle (of clk_ptp_ref_i) and the system time is updated whenever the accumulator overflows.

Offset 0x0B18 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TSAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TSAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 746. Timestamp Addend Register(MAC_TIMESTAMP_ADDEND)

Table 809. MAC_TIMESTAMP_ADDEND field descriptions

Field	Description
0:31 TSAR	Timestamp Addend Register This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

47.2.147 System Time Higher Word Seconds Register (MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS)

This register contains the most-significant 16-bits of timestamp seconds value.

Offset 0x0B1C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TSHWR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 747. System Time Higher Word Seconds Register (MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS)

Table 810. MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS field descriptions

Field	Description
16:31 TSHWR	Timestamp Higher Word Register This field contains the most-significant 16-bits of timestamp seconds value. This register is directly written to initialize the value and it is incremented when there is an overflow from 32-bits of the System Time Seconds register. Access restriction applies. Updated based on the event. Setting 1 sets. Setting 0 clears.

47.2.148 Timestamp Status Register (MAC_TIMESTAMP_STATUS)

All bits except Bits[4:6] gets cleared when the application reads this register.

Offset 0x0B20 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ATSNS					ATSSTM	0	0	0	0	0	0	ATSSTN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXTSSIS	0	0	0	0	0	0	0	0	0	0	0	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 748. Timestamp Status Register (MAC_TIMESTAMP_STATUS)

Table 811. MAC_TIMESTAMP_STATUS field descriptions

Field	Description
2:6 ATSNS	Number of Auxiliary Timestamp Snapshots This field indicates the number of Snapshots available in the FIFO. value equal to 4 indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.
7 ATSSTM	Auxiliary Timestamp Snapshot Trigger Missed This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

Table 811. MAC_TIMESTAMP_STATUS field descriptions (continued)

Field	Description
14:15 ATSSTN	Auxiliary Timestamp Snapshot Trigger Identifier These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. Bit 15: Auxiliary trigger 0 Bit 14: Auxiliary trigger 1 The software can read this register to find the triggers that are set when the timestamp is taken. Access restriction applies. Clears on read. Self-set to 1 on internal event.
16 TXTSSIS	Tx Timestamp Status Interrupt Status When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the MAC_TxTimestamp_Status_Nanoseconds and MAC_TxTimestamp_Status_Seconds registers. This bit is cleared when the MAC_TxTimestamp_Status_Seconds register is read.
28 TSTRGTERR 0	Timestamp Target Time Error This bit is set when the latest target time programed in the MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds registers elapses. This bit is cleared when the application reads this bit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 AUXTSTRIG	Auxiliary Timestamp Trigger Snapshot This bit is set high when the auxiliary snapshot is written to the FIFO. Access restriction applies. Clears on read. Self-set to 1 on internal event.
30 TSTARGT0	Timestamp Target Time Reached When set, this bit indicates that the value of system time is greater than or equal to the value specified in the MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds registers. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 TSSOVF	Timestamp Seconds Overflow When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.149 Transmit Timestamp Status Nanoseconds Register (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS)

This register contains the nanosecond part of timestamp captured for Transmit when Tx status is disabled.

The MAC_TxTimestamp_Status_Nanoseconds register, along with MAC_TxTimestamp_Status_Seconds, gives the 64-bit timestamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of MAC_TxTimestamp_Status_Nanoseconds is read. In the little-endian mode, this means when bits[31:24] are read; in big-endian mode, bits[7:0] are read.

If the application does not read these registers and timestamp of another packet is captured, then either the current timestamp is lost (overwritten) or the new timestamp is lost (dropped), depending on the setting of the TXTSSTSM bit of the MAC_Timestamp_Control register. The status bit TXTSC bit [16] in MAC_Timestamp_Status register is set whenever the MAC transmitter captures the timestamp.

Offset 0x0B30 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXTSSMIS	TXTSSLO														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXTSSLO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 749. Transmit Timestamp Status Nanoseconds Register (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS)

Table 812. MAC_TX_TIMESTAMP_STATUS_NANOSECONDS field descriptions

Field	Description
0 TXTSSMIS	Transmit Timestamp Status Missed When this bit is set, it indicates one of the following: The timestamp of the current packet is ignored if TXTSSTSM bit of the MAC_Timestamp_Control register is reset The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the MAC_Timestamp_Control register is set. Access restriction applies. Clears on read. Self-set to 1 on internal event.
1:31 TXTSSLO	Transmit Timestamp Status Low This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp.

47.2.150 Transmit Timestamp Status Seconds Register (MAC_TX_TIMESTAMP_STATUS_SECONDS)

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

Offset 0x0B34 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXTSSHI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXTSSHI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 750. Transmit Timestamp Status Seconds Register (MAC_TX_TIMESTAMP_STATUS_SECONDS)

Table 813. MAC_TX_TIMESTAMP_STATUS_SECONDS field descriptions

Field	Description
0:31 TXTSSHI	Transmit Timestamp Status High This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

47.2.151 Auxiliary Control Register (MAC_AUXILIARY_CONTROL)

The Auxiliary Timestamp Control register controls the Auxiliary Timestamp snapshot.

Offset 0x0B40 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ATSEN1	ATSEN0	0	0	0	ATSFC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 751. Auxiliary Control Register (MAC_AUXILIARY_CONTROL)

Table 814. MAC_AUXILIARY_CONTROL field descriptions

Field	Description
26 ATSEN1	Auxiliary Snapshot 1 Enable This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on ptp_aux_trig_i[1] input is enabled. When this bit is reset, the events on this input are ignored.
27 ATSEN0	Auxiliary Snapshot 0 Enable This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on ptp_aux_trig_i[0] input is enabled. When this bit is reset, the events on this input are ignored.
31 ATSFC	Auxiliary Snapshot FIFO Clear When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.152 Auxiliary Timestamp Nanoseconds Register (MAC_AUXILIARY_TIMESTAMP_NANOSECONDS)

The Auxiliary Timestamp Nanoseconds register, along with MAC_Auxiliary_Timestamp_Seconds, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4.

You can store multiple snapshots in this FIFO. Bits[2:6] in MAC_Timestamp_Status indicate the fill-level of the FIFO. The top of the FIFO is removed only when the last byte of MAC Register 91 (Auxiliary Timestamp - Seconds Register) is read. In the little-endian mode, this means when Bits[31:24] are read and in big-endian mode, Bits[7:0] are read.

Offset 0x0B48 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	AUXTSLO														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AUXTSLO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 752. Auxiliary Timestamp Nanoseconds Register (MAC_AUXILIARY_TIMESTAMP_NANOSECONDS)

Table 815. MAC_AUXILIARY_TIMESTAMP_NANOSECONDS field descriptions

Field	Description
1:31 AUXTSLO	Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

47.2.153 Auxiliary Timestamp Seconds Register (MAC_AUXILIARY_TIMESTAMP_SECONDS)

The Auxiliary Timestamp - Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

Offset 0x0B4C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AUXTSHI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AUXTSHI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 753. Auxiliary Timestamp Seconds Register (MAC_AUXILIARY_TIMESTAMP_SECONDS)

Table 816. MAC_AUXILIARY_TIMESTAMP_SECONDS field descriptions

Field	Description
0:31 AUXTSHI	Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

47.2.154 Timestamp Ingress Asymmetry Correction Register (MAC_TIMESTAMP_INGRESS_ASYM_CORR)

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

Offset 0x0B50 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	OSTIAC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	OSTIAC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 754. Timestamp Ingress Asymmetry Correction Register (MAC_TIMESTAMP_INGRESS_ASYM_CORR)

Table 817. MAC_TIMESTAMP_INGRESS_ASYM_CORR field descriptions

Field	Description
0:31 OSTIAC	One-Step Timestamp Ingress Asymmetry Correction This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programed value should be in units of nanoseconds and multiplied by 2^16. For example, 2.5 ns is represented as 0x00028000. The value can also be negative, which is represented in 2's complement form with bit 0 representing the sign bit.

47.2.155 Timestamp Egress Asymmetry Correction Register (MAC_TIMESTAMP_EGRESS_ASYM_CORR)

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

Offset 0x0B54 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	OSTEAC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	OSTEAC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 755. Timestamp Egress Asymmetry Correction Register (MAC_TIMESTAMP_EGRESS_ASYM_CORR)

Table 818. MAC_TIMESTAMP_EGRESS_ASYM_CORR field descriptions

Field	Description
0:31 OSTEAC	One-Step Timestamp Egress Asymmetry Correction This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programed value must be the negated value in units of nanoseconds multiplied by 2^16. For example, if the required correction is +2.5 ns, the programed value must be 0xFFFD_8000, which is the 2's complement of 0x0002_8000 (2.5 * 216). Similarly, if the required correction is -3.3 ns, the programed value is 0x0003_4CCC (3.3 * 216).

47.2.156 Timestamp Ingress Correction Nanosecond Register (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND)

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

Offset 0x0B58 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TSIC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TSIC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 756. Timestamp Ingress Correction Nanosecond Register (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND)

Table 819. MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND field descriptions

Field	Description
0:31 TSIC	Timestamp Ingress Correction This field contains the ingress path correction value as defined by the Ingress Correction expression.

47.2.157 Timestamp Egress Correction Nanosecond Register (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND)

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

Offset 0x0B5C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TSEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TSEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 757. Timestamp Egress Correction Nanosecond Register (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND)

Table 820. MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND field descriptions

Field	Description
0:31 TSEC	Timestamp Egress Correction This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

47.2.158 PPS Control Register (MAC_PPS_CONTROL)

Offset 0x0B70 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	TRGTMODSEL0		PPSEN0	PPSCTRL_PPSCMD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 758. PPS Control Register (MAC_PPS_CONTROL)

Table 821. MAC_PPS_CONTROL field descriptions

Field	Description
25:26 TRGTMODSEL0	Target Time Register Mode for PPS0 Output This field indicates the Target Time registers (MAC registers 96 and 97) mode for PPS0 output signal: 00 Target Time registers are programed only for generating the interrupt event. 01 Reserved 10 Target Time registers are programed for generating the interrupt event and starting or stopping the PPS0 output signal generation. 11 Target Time registers are programed only for starting or stopping the PPS0 output signal generation. No interrupt is asserted.

Table 821. MAC_PPS_CONTROL field descriptions (continued)

Field	Description
27 PPSEN0	Flexible PPS Output Mode Enable When this bit is set, Bits[28:31] function as PPSCMD. When this bit is reset, Bits[28:31] function as PPSCTRL (Fixed PPS mode).
28:31 PPSCTRL _PPSCMD	**PPS Output Frequency Control** This field controls the frequency of the PPS0 output (ptp_pps_o[0]) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies: 0001 The binary rollover is 2 Hz, and the digital rollover is 1 Hz 0010 The binary rollover is 4 Hz, and the digital rollover is 2 Hz 0011 The binary rollover is 8 Hz, and the digital rollover is 4 Hz 0100 The binary rollover is 16 Hz, and the digital rollover is 8 Hz ... 1111 The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz **Note:** In the binary rollover mode, the PPS output (ptp_pps_o) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second.(1) or **Flexible PPS Output (ptp_pps_o[0]) Control** Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programed only when they are 'all-zero'. The following list describes the values of PPSCMD0: - 0000 No Command - 0001 START Single Pulse. This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS0 Width Register. - 0010 START Pulse Train. This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS0 Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands. - 0011 Cancel START. This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programed start time. - 0100 STOP Pulse train at time. This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010) after the time programed in the Target Time registers elapses. - 0101 STOP Pulse Train immediately. This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010). - 0110 Cancel STOP Pulse train. This command cancels the STOP pulse train at time command if the programed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command. - 0111 Reserved - 1111 Reserved Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

1. For example:
When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms
When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of one clock of 50 percent duty cycle and 537 ms period. Second clock of 463 ms period (268 ms low and 195 ms high)
When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of three clocks of 50 percent duty cycle and 268 ms period. Fourth clock of 195 ms period (134 ms low and 61 ms high)
This behavior is because of the non-linear toggling of bits in the digital rollover mode in the MAC_System_Time_Nanoseconds register.

47.2.159 PPS0 Target Time Seconds Register (MAC_PPS0_TARGET_TIME_SECONDS)

The PPS Target Time Seconds register, along with PPS Target Time Nanoseconds register, is used to schedule an interrupt event [Bit 30 of MAC_Timestamp_Status] when the system time exceeds the value programed in these registers

Offset 0x0B80 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TSTRH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TSTRH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 759. PPS0 Target Time Seconds Register (MAC_PPS0_TARGET_TIME_SECONDS)

Table 822. MAC_PPS0_TARGET_TIME_SECONDS field descriptions

Field	Description
0:31 TSTRH0	PPS Target Time Seconds Register This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the MAC_PPS_Control register.

47.2.160 PPS0 Target Time Nanoseconds Register (MAC_PPS0_TARGET_TIME_NANOSECONDS)

Offset 0x0B84 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TRGTBUSY0	TTSL0														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TTSL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 760. PPS0 Target Time Nanoseconds Register (MAC_PPS0_TARGET_TIME_NANOSECONDS)

Table 823. MAC_PPS0_TARGET_TIME_NANOSECONDS field descriptions

Field	Description
0 TRGTBUSY0	PPS Target Time Register Busy The MAC sets this bit when the PPSCMD0 field in the MAC_PPS_Control register is programed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programed time gets corrupted.
1:31 TTSL0	Target Time Low for PPS Register This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [25:26]) in MAC_PPS_Control. When the TSCTRLSSR bit is set in the MAC_Timestamp_Control register, this value should not exceed 0x3B9A_C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.161 PPS0_Interval Register (MAC_PPS0_INTERVAL)

The PPS0 Interval register contains the number of units of sub-second increment value between the rising edges of PPS0 signal output (ptp_pps_o[0]).

Offset 0x0B88 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	PPSINT0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	PPSINT0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 761. PPS0_Interval Register (MAC_PPS0_INTERVAL)

Table 824. MAC_PPS0_INTERVAL field descriptions

Field	Description
0:31 PPSINT0	PPS Output Signal Interval These bits store the interval between the rising edges of PPS0 signal output. The interval is stored in terms of number of units of sub-second increment value. You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS0 signal output is 100 ns (that is, 5 units of sub-second increment value), you should program value 4 (5-1) in this register.

47.2.162 PPS0 Width Register (MAC_PPS0_WIDTH)

The PPS0 Width register contains the number of units of sub-second increment value between the rising and corresponding falling edges of PPS0 signal output (ptp_pps_o[0]).

Offset 0x0B8C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	PPSWIDTH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	PPSWIDTH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 762. PPS0 Width Register (MAC_PPS0_WIDTH)

Table 825. MAC_PPS0_WIDTH field descriptions

Field	Description
0:31 PPSWIDTH0	PPS Output Signal Width These bits store the width between the rising edge and corresponding falling edge of PPS0 signal output. The width is stored in terms of number of units of sub-second increment value. You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS0 signal output is 80 ns (that is, four units of sub-second increment value), you should program value 3 (4-1) in this register. **Note:** The value programed in this register must be lesser than the value programed in MAC_PPS0_Interval.

47.2.163 Operation Mode Register (MTL_OPERATION_MODE)

The Operation Mode register establishes the Transmit and Receive operating modes and commands.

Offset 0x0C00 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CNTCLR	CNTPRST	0	SCHALG		0	0	RAA	DTXSTS	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 763. Operation Mode Register (MTL_OPERATION_MODE)

Table 826. MTL_OPERATION_MODE field descriptions

Field	Description
22 CNTCLR	Counters Reset When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNT_PRESET bit, CNT_PRESET has precedence. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
23 CNTPRST	Counters Preset When this bit is set, MTL_TxQ[0-7]_Underflow register is initialized/preset to 12'h7F0. Missed Packet and Overflow Packet counters in MTL_RxQ[0-7]_Missed_Packet_Overflow_Cnt register is initialized/preset to 12'h7F0. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

Table 826. MTL_OPERATION_MODE field descriptions (continued)

Field	Description
25:26 SCHALG	Tx Scheduling Algorithm This field indicates the algorithm for Tx scheduling: 00 WRR algorithm 01 Reserved 10 Reserved. 11 Strict priority algorithm.
29 RAA	Receive Arbitration Algorithm This field is used to select the arbitration algorithm for the Rx side. 0 Strict priority (SP) Queue 0 has the lowest priority and the last queue has the highest priority. 1 Weighted Strict Priority (WSP)
30 DTXSTS	Drop Transmit Status When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

47.2.164 Debug Access Control Register (MTL_DBG_CTL)

The FIFO Debug Access Control and Status register controls the operation mode of FIFO debug access.

Offset 0x0C08 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	STSIE	PKTIE	FIFOSEL		FIFOWREN	FIFORDEN	RSTSEL	RSTALL	0	PKTSTATE		0	BYTEEN		DBGMOD	FDBGEN
W	STSIE	PKTIE	FIFOSEL		FIFOWREN	FIFORDEN	RSTSEL	RSTALL		PKTSTATE			BYTEEN		DBGMOD	FDBGEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 764. Debug Access Control Register (MTL_DBG_CTL)

Table 827. MTL_DBG_CTL field descriptions

Field	Description
16 STSIE	Transmit Status Available Interrupt Status Enable When this bit is set, an interrupt is generated when Transmit status is available in slave mode.
17 PKTIE	Receive Packet Available Interrupt Status Enable When this bit is set, an interrupt is generated when EOP of received packet is written to the Rx FIFO.
18:19 FIFOSEL	FIFO Selected for Access This field indicates the FIFO selected for debug access: 00 Tx FIFO 01 Tx Status FIFO (only read access when SLVMOD is set) 10 Reserved 11 Rx FIFO
20 FIFOWREN	FIFO Write Enable When this bit is set, it enables the Write operation on selected FIFO. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets.
21 FIFORDEN	FIFO Read Enable When this bit is set, it enables the Read operation on selected FIFO. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets.
22 RSTSEL	Reset Pointers of Selected FIFO When this bit is set, the pointers of the currently-selected FIFO are reset. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets.
23 RSTALL	Reset All Pointers When this bit is set, the pointers of all FIFOs are reset. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets.
25:26 PKTSTATE	Encoded Packet State This field is used to write the control information to the Tx FIFO or Rx FIFO. Tx FIFO / Rx FIFO: 00 Packet Data / Packet Data 01 Control Word/ Normal Status 10 SOP Data / Last Status 11 EOP Data / EOP
28:29 BYTEEN	Byte Enables This field indicates the number of data bytes valid in the data register during Write operation. This is valid only when PKTSTATE is 2'b10 (EOP) and Tx FIFO or Rx FIFO is selected. 00 Byte 0 valid 01 Byte 0 and Byte 1 are valid 10 Byte 0, Byte 1, and Byte 2 are valid 11 All four bytes are valid

Table 827. MTL_DBG_CTL field descriptions (continued)

Field	Description
30 DBGMOD	Debug Mode Access to FIFO When this bit is set, it indicates that the current access to the FIFO is read, write, and debug access. In this mode, the following access types are allowed: Read and Write access to Tx FIFO, TSO FIFO, and Rx FIFO Read access is allowed to Tx Status FIFO. When this bit is reset, it indicates that the current access to the FIFO is slave access bypassing the DMA. In this mode, the following access are allowed: Write access to the Tx FIFO Read access to the Rx FIFO and Tx Status FIFO
31 FDBGEN	FIFO Debug Access Enable When this bit is set, it indicates that the debug mode access to the FIFO is enabled. When this bit is reset, it indicates that the FIFO can be accessed only through a master interface.

47.2.165 Debug Status Register (MTL_DBG_STS)

The FIFO Debug Status register contains the status of FIFO debug access.

Offset 0x0C0C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LOCR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LOCR	0	0	0	0	0	STSI	PKTI	0	0	0	BYTEEN		PKTSTATE		FIFOBUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 765. Debug Status Register (MTL_DBG_STS)

Table 828. MTL_DBG_STS field descriptions

Field	Description
0:16 LOCR	Remaining Locations in the FIFO Slave Access Mode: This field indicates the space available in selected FIFO. Debug Access Mode: This field contains the Write or Read pointer value of the selected FIFO during Write or Read operation, respectively.
22 STSI	Transmit Status Available Interrupt Status When set, this bit indicates that the Slave mode Tx packet is transmitted, and the status is available in Tx Status FIFO. This bit is reset when 1 is written to this bit.
23 PKTI	Receive Packet Available Interrupt Status When set, this bit indicates that MAC layer has written the EOP of received packet to the Rx FIFO. This bit is reset when 1 is written to this bit.
27:28 BYTEEN	Byte Enables This field indicates the number of data bytes valid in the data register during Read operation. This is valid only when PKTSTATE is 2'b10 (EOP) and Tx FIFO or Rx FIFO is selected. 00 Byte 0 valid 01 Byte 0 and Byte 1 are valid 10 Byte 0, Byte 1, and Byte 2 are valid 11 All four bytes are valid
29:30 PKTSTATE	Encoded Packet State This field is used to get the control or status information of the selected FIFO. Tx FIFO: 00 Packet Data 01 Control Word 10 SOP Data 11 EOP Data Rx FIFO: 00 Packet Data 01 Normal Status 10 Last Status 11 EOP This field is applicable only for Tx FIFO and Rx FIFO during Read operation.
31 FIFOBUSY	FIFO Busy When set, this bit indicates that a FIFO operation is in progress in the MAC and content of the following fields is not valid: All other fields of this register All fields of the MTL_FIFO_Debug_Data register

47.2.166 FIFO Debug Data register (MTL_FIFO_DEBUG_DATA)

The FIFO Debug Data register contains the data to be written to or read from the FIFOs.

Offset 0x0C10 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	FDBGDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	FDBGDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 766. FIFO Debug Data register (MTL_FIFO_DEBUG_DATA)

Table 829. MTL_FIFO_DEBUG_DATA field descriptions

Field	Description
0:31 FDBGDATA	FIFO Debug Data During debug or slave access write operation, this field contains the data to be written to the Tx FIFO, Rx FIFO,. During debug or slave access read operation, this field contains the data read from the Tx FIFO, Rx FIFO, or Tx Status FIFO.

47.2.167 Interrupt Status Register (MTL_INTERRUPT_STATUS)

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

Offset 0x0C20 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DBGIS	MACIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Q1IS	Q0IS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 767. Interrupt Status Register (MTL_INTERRUPT_STATUS)

Table 830. MTL_INTERRUPT_STATUS field descriptions

Field	Description
14 DBGIS	Debug Interrupt status This bit indicates an interrupt event during the slave access. To reset this bit, the application must read the FIFO Debug Access Status register to get the exact cause of the interrupt and clear its source.
15 MACIS	MAC Interrupt Status This bit indicates an interrupt event in the MAC. To reset this bit, the application must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.
30 Q1IS	Queue 1 Interrupt status This bit indicates that there is an interrupt from Queue 1. To reset this bit, the application must read the MTL_Q1_Interrupt_Control_Status register to get the exact cause of the interrupt and clear its source.
31 Q0IS	Queue 0 Interrupt status This bit indicates that there is an interrupt from Queue 0. To reset this bit, the application must read Queue 0 Interrupt Control and Status register to get the exact cause of the interrupt and clear its source.

47.2.168 Receive Queue and DMA Channel Mapping 0 Register (MTL_RXQ_DMA_MAP0)

The Receive Queue and DMA Channel Mapping 0 register.

Offset 0x0C30 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	Q1DDMACH	0	0	0	Q1MDMACH	0	0	0	Q0DDMACH	0	0	0	Q0MDMACH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 768. Receive Queue and DMA Channel Mapping 0 Register (MTL_RXQ_DMA_MAP0)

Table 831. MTL_RXQ_DMA_MAP0 field descriptions

Field	Description
19 Q1DDMACH	Queue 1 Enabled for DA-based DMA Channel Selection When set, this bit indicates that the packets received in Queue 1 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 1 are routed to the DMA Channel programed in the Q1MDMACH field (Bits[21:23]).
23 Q1MDMACH	Queue 1 Mapped to DMA Channel This field controls the routing of the received packet in Queue 1 to the DMA channel: 0 DMA Channel 0 1 DMA Channel 1 This field is valid when the Q1DDMACH field is reset.
27 Q0DDMACH	Queue 0 Enabled for DA-based DMA Channel Selection When set, this bit indicates that the packets received in Queue 0 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 0 are routed to the DMA Channel programed in the Q0MDMACH field.
31 Q0MDMACH	Queue 0 Mapped to DMA Channel This field controls the routing of the packet received in Queue 0 to the DMA channel: 0 DMA Channel 0 1 DMA Channel 1 This field is valid when the Q0DDMACH field is reset.

47.2.169 Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE)

The Queue 0 Transmit Operation Mode register establishes the Transmit queue operating modes and commands.

Offset 0x0D00 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TQS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	TTC			TXQEN		TSF	FTQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 769. Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE)

Table 832. MTL_TXQ0_OPERATION_MODE field descriptions

Field	Description
12:15 TQS	Transmit Queue Size This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The sixteenth bit is the starting bit of this field. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 4096, the width of this field is 4 bits: LOG2(4096/256) = LOG2(16) = 4 bits A value of 0 indicates 256 bytes
25:27 TTC	Transmit Threshold Control These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. 000 32 001 64 010 96 011 128 100 192 101 256 110 384 111 512
28:29 TXQEN	Transmit Queue Enable This field is used to enable/disable the transmit queue 0. 00 Not enabled 01 Reserved 10 Enabled 11 Reserved
30 TSF	Transmit Store and Forward When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[25:27] of this register are ignored. This bit should be changed only when the transmission is stopped.
31 FTQ	Flush Transmit Queue When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the MTL_TxQ1_Operation_Mode register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. **Note:** The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.170 Queue 0 Underflow Counter Register (MTL_TXQ0_UNDERFLOW)

The Queue 0 Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush.

Offset 0x0D04 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	UFCNTOVF	UFFRMCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 770. Queue 0 Underflow Counter Register (MTL_TXQ0_UNDERFLOW)

Table 833. MTL_TXQ0_UNDERFLOW field descriptions

Field	Description
20 UFCNTOVF	Overflow Bit for Underflow Packet Counter This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 UFFRMCNT	Underflow Packet Counter This field indicates the number of packets aborted by the controller because of Tx Queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.171 Queue 0 Transmit Debug Register (MTL_TXQ0_DEBUG)

The Queue 0 Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

Offset 0x0D08 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	STXSTSF			0	PTXQ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS		TXQPAUSED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 771. Queue 0 Transmit Debug Register (MTL_TXQ0_DEBUG)

Table 834. MTL_TXQ0_DEBUG field descriptions

Field	Description
9:11 STXSTSF	Number of Status Words in Tx Status FIFO of Queue This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.
13:15 PTXQ	Number of Packets in the Transmit Queue This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of packets in the Transmit queue.
26 TXSTSFSTS	MTL Tx Status FIFO Full Status When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.
27 TXQSTS	MTL Tx Queue Not Empty Status When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission.
28 TWCSTS	MTL Tx Queue Write Controller Status When high, this bit indicates that the MTL Tx Queue Write Controller is active, and it is transferring the data to the Tx Queue.

Table 834. MTL_TXQ0_DEBUG field descriptions (continued)

Field	Description
29:30 TRCSTS	MTL Tx Queue Read Controller Status This field indicates the state of the Tx Queue Read Controller: 00 Idle state 01 Read state (transferring data to the MAC transmitter) 10 Waiting for pending Tx Status from the MAC transmitter 11 Flushing the Tx queue because of the Packet Abort request from the MAC
31 TXQPAUSED	Transmit Queue in Pause When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the Pause condition (in the full-duplex only mode) because of the following: Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled Reception of 802.3x Pause packet when PFC is disabled

47.2.172 Queue 0 ETS Status Register (MTL_TXQ0_ETS_STATUS)

The Queue 0 ETS Status register provides the average traffic transmitted in Queue 0.

Offset 0x0D14 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	ABS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ABS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 772. Queue 0 ETS Status Register (MTL_TXQ0_ETS_STATUS)

Table 835. MTL_TXQ0_ETS_STATUS field descriptions

Field	Description
8:31 ABS	Average Bits per Slot This field contains the average transmitted bits per slot.

47.2.173 Queue 0 Quantum or Weights Register (MTL_TXQ0_QUANTUM_WEIGHT)

The Queue 0 Quantum or Weights register contains weights for the Weighted Round Robin (WRR), for Queue 0.

Offset 0x0D18 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	ISCQW				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ISCQW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 773. Queue 0 Quantum or Weights Register (MTL_TXQ0_QUANTUM_WEIGHT)

Table 836. MTL_TXQ0_QUANTUM_WEIGHT field descriptions

Field	Description
11:31 ISCQW	When DWRR algorithm is enabled for Queue 0 traffic, this field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. When WRR algorithm is enabled for Queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [11:24] must be written to zero.

47.2.174 Interrupt Control Status Register (MTL_Q0_INTERRUPT_CONTROL_STATUS)

This register contains the interrupt enable and status bits for the queue 0 interrupts.

Offset 0x0D2C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RXOIE	0	0	0	0	0	0	0	RXOVFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ABPSIE	TXUIE	0	0	0	0	0	0	ABPSIS	TXUNFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 774. Interrupt Control Status Register (MTL_Q0_INTERRUPT_CONTROL_STATUS)

Table 837. MTL_Q0_INTERRUPT_CONTROL_STATUS field descriptions

Field	Description
7 RXOIE	Receive Queue Overflow Interrupt Enable When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.
15 RXOVFIS	Receive Queue Overflow Interrupt Status This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[10]. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
22 ABPSIE	Average Bits Per Slot Interrupt Enable When this bit is set, the MAC asserts the interrupt or mci_intr_o interrupt when the average bits per slot status is updated. When this bit is cleared, the interrupt is not asserted for such an event.
23 TXUIE	Transmit Queue Underflow Interrupt Enable When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.
30 ABPSIS	Average Bits Per Slot Interrupt Status When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
31 TXUNFIS	Transmit Queue Underflow Interrupt Status This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

47.2.175 Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE)

The Queue 0 Receive Operation Mode register establishes the Receive queue operating modes and command.

Offset 0x0D30 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RQS					0	0	RFD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFD		0	0	RFA				EHFC	DIS_TCP_EF	RSF	FEP	FUP	0	RTC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 775. Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE)

Table 838. MTL_RXQ0_OPERATION_MODE field descriptions

Bits	Description
7:11 RQS	Receive Queue Size This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx Queues more than one and the reset value is 0x0. When the number of Rx Queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Rx memory size. For example, if the memory size is 8192, the width of this field is 5 bits: LOG2(8192/256) = LOG2(32) = 5 bits
14:17 RFD	Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes) These bits control the threshold (fill-level of Rx queue) at which the flow control is de-asserted after activation: 0 Full minus 1 KB, that is, FULL 1 KB 1 Full minus 1.5 KB, that is, FULL 1.5 KB 2 Full minus 2 KB, that is, FULL 2 KB 3 Full minus 2.5 KB, that is, FULL 2.5 KB ... 62 Full minus 32 KB, that is, FULL 32 KB 63 Full minus 32.5 KB, that is, FULL 32.5 KB The de-assertion is effective only after flow control is asserted. Note: The value must be programed in such a way to make sure that the threshold is a positive number. When the EHFC is set high, these values are applicable only when the Rx queue size determined by the RQS field of this register, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the RQS field of this register.

Table 838. MTL_RXQ0_OPERATION_MODE field descriptions (continued)

Bits	Description
20:23 RFA	Threshold for Activating Flow Control (in half-duplex and full-duplex These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, refer to RFD.
24 EHFC	Enable Hardware Flow Control When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled.
25 DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset.
26 RSF	Receive Queue Store and Forward When this bit is set, the Ethernet module reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.
27 FEP	Forward Error Packets When this bit is reset, the Rx queue drops packets with error status (CRC error, MII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.
28 FUP	Forward Undersized Good Packets When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.
30:31 RTC	Receive Queue Threshold Control These bits control the threshold level of the MTL Rx queue (in bytes): 00 64 01 32 10 96 11 128 The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

47.2.176 Queue 0 Missed Packet and Overflow Counter Register (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT)

The Queue 0 Missed Packet and Overflow Counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

Offset 0x0D34 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	MISCNTOVF	MISPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	OVFCNTOVF	OVFPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 776. Queue 0 Missed Packet and Overflow Counter Register (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT)

Table 839. MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT field descriptions

Field	Description
4 MISCNTOVF	Missed Packet Counter Overflow Bit When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5:15 MISPKTCNT	Missed Packet Counter This field indicates the number of packets missed by the Ethernet module because the application asserted ari_pkt_flush_i[] for this queue. This counter is incremented each time the application issues ari_pkt_flush_i[] for this queue. This counter is reset when this register is read with mci_be_i[0] at 1b1. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 839. MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT field descriptions

Field	Description
20 OVFCNTOVF	Overflow Counter Overflow Bit When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 OVFPKTCNT	Overflow Packet Counter This field indicates the number of packets discarded by the Ethernet module because of Receive queue overflow. This counter is incremented each time the Ethernet module discards an incoming packet because of overflow. This counter is reset when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.177 Queue 0 Receive Debug Register (MTL_RXQ0_DEBUG)

The Queue 0 Receive Debug register gives the debug status of various blocks related to the Receive queue.

Offset 0x0D38 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PRXQ													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	RXQSTS		0	RRCSTS		RWCSTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 777. Queue 0 Receive Debug Register (MTL_RXQ0_DEBUG)

Table 840. MTL_RXQ0_DEBUG field descriptions

Field	Description
2:15 PRXQ	Number of Packets in Receive Queue This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.
26:27 RXQSTS	MTL Rx Queue Fill-Level Status This field gives the status of the fill-level of the Rx Queue: 00 Rx Queue empty 01 Rx Queue fill-level below flow-control deactivate threshold 10 Rx Queue fill-level above flow-control activate threshold 11 Rx Queue full

Table 840. MTL_RXQ0_DEBUG field descriptions (continued)

Field	Description
29:30 RRCSTS	MTL Rx Queue Read Controller State This field gives the state of the Rx queue Read controller: 00 Idle state 01 Reading packet data 10 Reading packet status (or timestamp) 11 Flushing the packet data and status
31 RWCSTS	MTL Rx Queue Write Controller Active Status When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx Queue.

47.2.178 Queue Receive Control Register (MTL_RXQ0_CONTROL)

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

Offset 0x0D3C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXQ_FRM_ARBIT	RXQ_WEGT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 778. Queue Receive Control Register (MTL_RXQ0_CONTROL)

Table 841. MTL_RXQ0_CONTROL field descriptions

Field	Description
28 RXQ_FRM_ARBIT	Receive Queue Packet Arbitration When this bit is set, the Ethernet module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the Ethernet module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: PBL amount of data (indicated by ari_qN_pbl_i[]) or complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the Ethernet module drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode).
29:31 RXQ_WEGT	Receive Queue Weight This field indicates the weight assigned to the Rx Queue 0. The weight is used as the number of continuous PBL requests or contiguous packets (depending on the RXQ_PKT_ARBIT) allocated to the queue in one arbitration cycle.

47.2.179 Transmit Q*n* Operation Mode register (MTL_TXQ*n*_OPERATION_MODE)

The Queue 1 Transmit Operation Mode register establishes the Transmit queue operating modes and commands.

Offset 0x0D00 + *n**0x40 (n=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TQS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	TTC			TXQEN		TSF	FTQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 779. Transmit Q*n* Operation Mode register (MTL_TXQ*n*_OPERATION_MODE)

Table 842. MTL_TXQ*n*_OPERATION_MODE field descriptions

Field	Description
12:15 TQS	Transmit Queue Size This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The sixteenth bit is the starting bit of this field. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(4096/256) = LOG2(16) = 4 bits
25:27 TTC	Transmit Threshold Control These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. 000 32 001 64 010 96 011 128 100 192 101 256 110 384 111 512
28:29 TXQEN	Transmit Queue Enable This field is used to enable/disable the transmit queue n. 00 Not enabled 01 Enable in AV mode 10 Enabled 11 Reserved
30 TSF	Transmit Store and Forward When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[25:27] of this register are ignored. This bit should be changed only when the transmission is stopped.
31 FTQ	Flush Transmit Queue When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the MTL_TxQ1_Operation_Mode register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. **Note:** The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.180 Transmit Q*n* Underflow register (MTL_TXQ*n*_UNDERFLOW)

The Queue 1 Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush.

Offset 0x0D04 + *n**0x40 (n=1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	UFCNTOVF	UFFRMCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 780. Transmit Q*n* Underflow register (MTL_TXQ*n*_UNDERFLOW)

Table 843. MTL_TXQ*n*_UNDERFLOW field descriptions

Field	Description
20 UFCNTOVF	Overflow Bit for Underflow Packet Counter This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 UFFRMCNT	Underflow Packet Counter This field indicates the number of packets aborted by the controller because of Tx Queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.181 Transmit Q*n* Debug register (MTL_TXQ*n*_DEBUG)

The Queue 1 Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

Offset 0x0D08 + *n**0x40 (*n*=1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	STXSTSF			0	PTXQ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS		TXQPAUSED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 781. Transmit Q Debug register (MTL_TXQ*n*_DEBUG)

Table 844. MTL_TXQ*n*_DEBUG field descriptions

Field	Description
9:11 STXSTSF	Number of Status Words in Tx Status FIFO of Queue This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.
13:15 PTXQ	Number of Packets in the Transmit Queue This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of packets in the Transmit queue.
26 TXSTSFSTS	MTL Tx Status FIFO Full Status When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.
27 TXQSTS	MTL Tx Queue Not Empty Status When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission.
28 TWCSTS	MTL Tx Queue Write Controller Status When high, this bit indicates that the MTL Tx Queue Write Controller is active, and it is transferring the data to the Tx Queue.

Table 844. MTL_TXQ*n*_DEBUG field descriptions (continued)

Field	Description
29:30 TRCSTS	MTL Tx Queue Read Controller Status This field indicates the state of the Tx Queue Read Controller: 00 Idle state 01 Read state (transferring data to the MAC transmitter) 10 Waiting for pending Tx Status from the MAC transmitter 11 Flushing the Tx queue because of the Packet Abort request from the MAC
31 TXQPAUSED	Transmit Queue in Pause When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the Pause condition (in the full-duplex only mode) because of the following: Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled Reception of 802.3x Pause packet when PFC is disabled

47.2.182 Transmit Q*n* ETS Control register (MTL_TXQ*n*_ETS_CONTROL)

The Queue ETS Control register controls the enhanced transmission selection operation.

Offset 0x0D10 + *n**0x40 (*n*=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	SLC			CC	AVALG	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 782. MTL_Transmit Q*n* ETS Control register (MTL_TXQ*n*_ETS_CONTROL)

Table 845. MTL_TXQ*n*_ETS_CONTROL field descriptions

Field	Description
25:27 SLC	Slot Count If the credit-based shaper algorithm is enabled, the software can program the number of slots (of 125 µs duration) over which the average transmitted bits per slot, provided in the MTL_TxQ1_ETS_Status register, need to be computed for Queue 1. The encoding is as follows: 000 1 Slot 001 2 Slots 010 4 Slots 011 8 Slots 100 16 Slots 101 Reserved 110 Reserved 111 Reserved
28 CC	Credit Control When this bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in Channel 1. The credit accumulates even when there is no packet waiting in Channel 1 and another channel is transmitting. When this bit is reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in Channel 1. When there is no packet waiting in Channel 1 and other channel is transmitting, no credit is accumulated.
29 AVALG	AV Algorithm When Queue 1 is programed for AV, this field configures the scheduling algorithm for this queue: This bit when set, indicates credit based shaper algorithm (CBS) is selected for Queue 1 traffic. When reset, strict priority is selected.

47.2.183 Transmit Q*n* ETS Status register (MTL_TXQ*n*_ETS_STATUS)

The Queue 1 ETS Status register provides the average traffic transmitted in Queue 1.

Offset 0x0D14 + *n**0x40 (n=1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	ABS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ABS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 783. Transmit Q*n* ETS Status register (MTL_TXQ*n*_ETS_STATUS)

Table 846. MTL_TXQ*n*_ETS_STATUS field descriptions

Field	Description
8:31 ABS	Average Bits per Slot This field contains the average transmitted bits per slot. For AV operation for Queue 1, this field is computed over number of slots, programed in the SLC field of MTL_TxQ1_ETS_CONTROL register, for Queue 1 traffic. The maximum value is 0x30D4 for 100 Mbps and 0x1E848 for 1000 Mbps. Bits [8:14] are always read as zero.

47.2.184 Transmit Q*n* Quantum Weight register (MTL_TXQ*n*_QUANTUM_WEIGHT)

The Queue 1 idleSlopeCredit, Quantum or Weights register provides the average traffic transmitted in Queue 1.

Offset 0x0D18 + *n**0x40 (n=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	ISCQW				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ISCQW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 784. Transmit Q*n* Quantum Weight register (MTL_TXQ*n*_QUANTUM_WEIGHT)

Table 847. MTL_TXQ*n*_QUANTUM_WEIGHT field descriptions

Filed	Description
11:31 ISCQW	idleSlopeCredit, Quantum or Weights **idleSlopeCredit** When AV feature is enabled, this field contains the idleSlopeCredit value required for the credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode. Bits[20:14] must be written to zero. **Weights** When enabled with WRR algorithm for Queue 1 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [11:24] must be written to zero.

47.2.185 Transmit Q*n* Send Slope Credit register (MTL_TXQ*n*_SENDSPLOPECREDIT)

The sendSlopeCredit register contains the sendSlope credit value required for the credit-based shaper algorithm for the Queue.

Offset 0x0D1C + *n**0x40 (n=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	SSC													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 785. Transmit Q*n* Send Slope Credit register (MTL_TXQ*n*_SENDSPLOPECREDIT)

Table 848. MTL_TXQ*n*_SENDSPLOPECREDIT field descriptions

Field	Description
18:31 SSC	sendSlopeCredit With AV operation enabled, this field contains the sendSlopeCredit value required for credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps) when the credit is decreasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x1000 in 100 Mbps mode. This field should be programed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when Channel 1 is selected for transmission.

47.2.186 Transmit Q*n* High Credit register (MTL_TXQ*n*_HICREDIT)

The hiCredit register contains the hiCredit value required for the credit-based shaper algorithm for the Queue.

Offset 0x0D20 + *n**0x40 (*n*=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	HC												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 786. MTL_Transmit Q*n* High Credit register (MTL_TXQ*n*_HICREDIT)

Table 849. MTL_TXQ*n*_HICREDIT field descriptions

Field	Description
3:31 HC	hiCredit When the AV feature is enabled, this field contains the hiCredit value required for the credit-based shaper algorithm. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16,384 bytes or 131,072 bits). The value to be specified is 131,072 * 1,024 = 134,217,728 or 0x0800_0000.

47.2.187 Transmit Q*n* Low Credit register (MTL_TXQ*n*_LOCREDIT)

The loCredit register contains the loCredit value required for the credit-based shaper algorithm for the Queue.

Offset 0x0D24 + *n**0x40 (*n*=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LC												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	LC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 787. Transmit Q*n* Low Credit register (MTL_TXQ*n*_LOCREDIT)

Table 850. MTL_TXQ*n*_LOCREDIT field descriptions

Field	Description
3:31 LC	loCredit When AV operation is enabled, this field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value is maxFrameSize transmitted from this queue, that is, 16384 * 8 * 1024 = 134,217,728 or 0x0800_0000. Because it is a negative value, the programed value is 2's complement of the value, that is, 0x1800_0000.

47.2.188 Q*n* Interrupt Control Status register (MTL_Q*n*_INTERRUPT_CONTROL_STATUS)

This register contains the interrupt enable and status bits for the queue 1 interrupts.

Offset 0x0D2C + *n**0x40 (n=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RXOIE								RXOVFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ABPSIE	TXUIE							ABPSIS	TXUNFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 788. Q*n* Interrupt Control Status register (MTL_Q*n*_INTERRUPT_CONTROL_STATUS)

Table 851. MTL_Q*n*_INTERRUPT_CONTROL_STATUS field descriptions

Field	Description
7 RXOIE	Receive Queue Overflow Interrupt Enable When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.
15 RXOVFIS	Receive Queue Overflow Interrupt Status This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
22 ABPSIE	Average Bits Per Slot Interrupt Enable When this bit is set, the MAC asserts the interrupt or mci_intr_o interrupt when the average bits per slot status is updated. When this bit is cleared, the interrupt is not asserted for such an event.
23 TXUIE	Transmit Queue Underflow Interrupt Enable When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.
30 ABPSIS	Average Bits Per Slot Interrupt Status When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
31 TXUNFIS	Transmit Queue Underflow Interrupt Status This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

47.2.189 Receive Q*n* Operation Mode register (MTL_RXQ*n*_OPERATION_MODE)

The Queue 1 Receive Operation Mode register establishes the Receive queue operating modes and command.

Offset 0x0D30 + *n**0x40 (*n*=1)

Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RQS							RFD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFD		0	0	RFA				EHFC	DIS_TCP_EF	RSF	FEP	FUP	0	RTC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 789. MTL_Receive Q*n* Operation Mode register (MTL_RXQ*n*_OPERATION_MODE)

Table 852. MTL_RXQ*n*_OPERATION_MODE field descriptions

Field	Description
7:11 RQS	Receive Queue Size This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx Queues more than one and the reset value is 0x0. When the number of Rx Queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Rx memory size selected in your configuration. For example, if the memory size is 8192, the width of this field is 5 bits: LOG2(8192/256) = LOG2(32) = 5 bits
14:17 RFD	Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes) These bits control the threshold (fill-level of Rx queue) at which the flow control is de-asserted after activation: 0 Full minus 1 KB, that is, FULL 1 KB 1 Full minus 1.5 KB, that is, FULL 1.5 KB 2 Full minus 2 KB, that is, FULL 2 KB 3 Full minus 2.5 KB, that is, FULL 2.5 KB ... 62 Full minus 32 KB, that is, FULL 32 KB 63 Full minus 32.5 KB, that is, FULL 32.5 KB The de-assertion is effective only after flow control is asserted. Note: The value must be programed in such a way to make sure that the threshold is a positive number. When the EHFC is set high, these values are applicable only when the Rx queue size determined by the RQS field of this register, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the RQS field of this register.
20:23 RFA	Threshold for Activating Flow Control (in half-duplex and full-duplex These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, refer to RFD.
24 EHFC	Enable Hardware Flow Control When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled.
25 DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset.
26 RSF	Receive Queue Store and Forward When this bit is set, the Ethernet module reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Table 852. MTL_RXQ*n*_OPERATION_MODE field descriptions (continued)

Field	Description
27 FEP	Forward Error Packets When this bit is reset, the Rx queue drops packets with error status (CRC error, MII_RX_ERR, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.
28 FUP	Forward Undersized Good Packets When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.
30:31 RTC	Receive Queue Threshold Control These bits control the threshold level of the MTL Rx queue (in bytes): 00 64 01 32 10 96 11 128 The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

47.2.190 Receive Q*n* Missed Packet and Overflow Counter register (MTL_RXQ*n*_MISSED_PACKET_OVERFLOW_CNT)

The Queue 1 Missed Packet and Overflow Counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

Offset 0x0D34 + *n**0x40 (*n*=1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	MISCNTOVF	MISPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	OVFCNTOVF	OVFPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 790. Receive Q*n* Missed Packet and Overflow Counter register (MTL_RXQ*n*_MISSED_PACKET_OVERFLOW_CNT)

Table 853. MTL_RXQ*n*_MISSED_PACKET_OVERFLOW_CNT field descriptions

Field	Description
4 MISCNTOVF	Missed Packet Counter Overflow Bit When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5:15 MISPKTCNT	Missed Packet Counter This field indicates the number of packets missed by the Ethernet module because the application asserted ari_pkt_flush_i[] for this queue. This counter is incremented each time the application issues ari_pkt_flush_i[] for this queue. This counter is reset when this register is read with mci_be_i[0] at 1b1. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.
20 OVFCNTOVF	Overflow Counter Overflow Bit When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 OVFPKTCNT	Overflow Packet Counter This field indicates the number of packets discarded by the Ethernet module because of Receive queue overflow. This counter is incremented each time the Ethernet module discards an incoming packet because of overflow. This counter is reset when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.2.191 Receive Q*n* Debug register (MTL_RXQ*n*_DEBUG)

The Queue 1 Receive Debug register gives the debug status of various blocks related to the Receive queue.

Offset 0x0D38 + *n**0x40 (n=1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PRXQ													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	RXQSTS		0	RRCSTS		RWCSTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 791. Receive Q*n* Debug register (MTL_RXQ*n*_DEBUG)

Table 854. MTL_RXQ*n*_DEBUG field descriptions

Field	Description
2:15 PRXQ	Number of Packets in Receive Queue This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.
26:27 RXQSTS	MTL Rx Queue Fill-Level Status This field gives the status of the fill-level of the Rx Queue: 00 Rx Queue empty 01 Rx Queue fill-level below flow-control deactivate threshold 10 Rx Queue fill-level above flow-control activate threshold 11 Rx Queue full
29:30 RRCSTS	MTL Rx Queue Read Controller State This field gives the state of the Rx queue Read controller: 00 Idle state 01 Reading packet data 10 Reading packet status (or timestamp) 11 Flushing the packet data and status
31 RWCSTS	MTL Rx Queue Write Controller Active Status When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx Queue.

47.2.192 Receive Q*n* Control register (MTL_RXQ*n*_CONTROL)

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

Offset 0x0D3C + *n**0x40 (*n*=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXQ_FRM_ARBIT	RXQ_WEGT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 792. Receive Q*n* Control register (MTL_RXQ*n*_CONTROL)

Table 855. MTL_RXQ*n*_CONTROL field descriptions

Field	Description
28 RXQ_FRM_ARBIT	Receive Queue Packet Arbitration When this bit is set, the Ethernet module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the Ethernet module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: PBL amount of data (indicated by ari_qN_pbl_i[]) or complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the Ethernet module drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode).
29:31 RXQ_WEGT	Receive Queue Weight This field indicates the weight assigned to the Rx Queue 0. The weight is used as the number of continuous PBL requests or contiguous packets (depending on the RXQ_PKT_ARBIT) allocated to the queue in one arbitration cycle.

47.2.193 DMA_Mode Register (DMA_MODE)

The Bus Mode register establishes the bus operating modes for the DMA.

Offset 0x1000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INTM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PR			TXPR	0	0	0	0	0	0	TAA			DA	SWR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 793. DMA_Mode Register (DMA_MODE)

Table 856. DMA_MODE field descriptions

Field	Description
14:15 INTM	Interrupt Mode This field defines the interrupt mode of the Ethernet module. The behavior of the following outputs changes depending on the following settings: sbd_perch_tx_intr_o[] (Transmit Per Channel Interrupt) sbd_perch_rx_intr_o[] (Receive Per Channel Interrupt) interrupt (Common Interrupt) It also changes the behavior of the RI/TI bits in the DMA_CH0_Status.
17:19 PR	Priority Ratio These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set. 000 The priority ratio is 1:1 001 The priority ratio is 2:1 010 The priority ratio is 3:1 011 The priority ratio is 4:1 100 The priority ratio is 5:1 101 The priority ratio is 6:1 110 The priority ratio is 7:1 111 The priority ratio is 8:1
20 TXPR	Transmit Priority When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.
27:29 TAA	Transmit Arbitration Algorithm This field is used to select the arbitration algorithm for the Transmit side when multiple Tx DMAs are selected. 000 Fixed priority In fixed priority, Channel 0 has the lowest priority and the last channel has the highest priority. 001 Weighted Strict Priority (WSP) 010 Weighted Round-Robin (WRR) 011 Reserved ... 111 Reserved

Table 856. DMA_MODE field descriptions (continued)

Field	Description
30 DA	DMA Tx or Rx Arbitration Scheme This bit specifies the arbitration scheme between the Transmit and Receive paths of all channels: 0 Weighted Round-Robin with Rx:Tx or Tx:Rx The priority between the paths is according to the priority specified in Bits[17:19] and the priority weight is specified in the TXPR bit. 1 Fixed Priority The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path.
31 SWR	Software Reset When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all Ethernet module clock domains. Before reprogramming any Ethernet module register, a value of zero should be read in this bit. **Note:** The reset operation is complete only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

47.2.194 System Bus Mode Register (DMA_SYSBUS_MODE)

The System Bus mode register controls the behavior of the AHB master. It mainly controls burst splitting and number of outstanding requests.

Offset 0x1004 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RB	MB	0	AAL	0	0	0	0	0	0	0	0	0	0	0	FB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 794. System Bus Mode Register (DMA_SYSBUS_MODE)

Table 857. DMA_SYSBUS_MODE field descriptions

Field	Description
16 RB	Rebuild INCRx Burst When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.
17 MB	Mixed Burst When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).
19 AAL	Address-Aligned Beats When this bit is set to 1, AHB master performs address-aligned burst transfers on Read and Write channels.
31 FB	Fixed Burst Length When this bit is set to 1, the AHB master will initiate burst transfers of specified length (INCRx or SINGLE). When this bit is set to 0, the AHB master will initiate transfers of unspecified length (INCR) or SINGLE transfers.

47.2.195 Interrupt Status Register (DMA_INTERRUPT_STATUS)

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

Offset 0x1008 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MACIS	MTLIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DC1IS	DC0IS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 795. Interrupt Status Register (DMA_INTERRUPT_STATUS)

Table 858. DMA_INTERRUPT_STATUS field descriptions

Field	Description
14 MACIS	MAC Interrupt Status This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.
15 MTLIS	MTL Interrupt Status This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.
30 DC1IS	DMA Channel 1 Interrupt Status This bit indicates an interrupt event in DMA Channel 1. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 1 to get the exact cause of the interrupt and clear its source.
31 DC0IS	DMA Channel 0 Interrupt Status This bit indicates an interrupt event in DMA Channel 0. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 0 to get the exact cause of the interrupt and clear its source.

47.2.196 Debug Status 0 Register (DMA_DEBUG_STATUS0)

The Debug Status 0 register gives the Receive and Transmit process status for DMA Channel 0-Channel 1 for debugging purpose.

Offset 0x100C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TPS1				RPS1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TPS0				RPS0				0	0	0	0	0	0	0	AXWHSTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 796. Debug Status 0 Register (DMA_DEBUG_STATUS0)

Table 859. DMA_DEBUG_STATUS0 field descriptions

Field	Description
8:11 TPS1	DMA Channel 1 Transmit Process State This field indicates the Tx DMA FSM state for Channel 1. This field is similar to the TPS0 field.
12:15 RPS1	DMA Channel 1 Receive Process State This field indicates the Rx DMA FSM state for Channel 1. This field is similar to the RPS0 field.
16:19 TPS0	DMA Channel 0 Transmit Process State This field indicates the Tx DMA FSM state for Channel 0: 000 Stopped (Reset or Stop Transmit Command issued) 001 Running (Fetching Tx Transfer Descriptor) 010 Running (Waiting for status) 011 Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO)) 100 Timestamp write state 101 Reserved for future use 110 Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow) 111 Running (Closing Tx Descriptor) This field does not generate an interrupt.
20:23 RPS0	DMA Channel 0 Receive Process State This field indicates the Rx DMA FSM state for Channel 0: 000 Stopped (Reset or Stop Receive Command issued) 001 Running (Fetching Rx Transfer Descriptor) 010 Reserved for future use 011 Running (Waiting for Rx packet) 100 Suspended (Rx Descriptor Unavailable) 101 Running (Closing the Rx Descriptor) 110 Timestamp write state 111 Running (Transferring the received packet data from the Rx buffer to the system memory) The MSB of this field always returns 0. This field does not generate an interrupt.
31 AXWHSTS	AHB Master Status When high, this bit indicates that the AHB master FSMs are in the non-idle state.

47.2.197 DMA Channel *n* Control register (DMA_CH*n*_CONTROL)

The DMA Channel*n* Control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as 8xPBL mode.

Offset 0x1100 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	DSL			0	PBLx8
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 797. DMA Channel *n* Control register (DMA_CH*n*_CONTROL)

Table 860. DMA_CH*n*_CONTROL field descriptions

Field	Description
11:13 DSL	Descriptor Skip Length This bit specifies the Dword number (64-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor. When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.
15 PBLx8	8xPBL mode When this bit is set, the PBL value programed in Bits[10:15] in DMA_CH0_Tx_Control is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

47.2.198 DMA Channel *n* Transmit Control register (DMA_CH*n*_TX_CONTROL)

The DMA Channel Transmit Control register controls the Tx features such as PBL, and Tx Channel weights.

Offset 0x1104 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TxPBL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	OSF	TCW			ST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 798. DMA Channel *n* Transmit Control register (DMA_CH*n*_TX_CONTROL)

Table 861. DMA_CH*n*_TX_CONTROL field descriptions

Field	Description
10:15 TxPBL	Transmit Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: – Set the PBLx8 mode in DMA_CH0_Control register. – Set the PBL.
27 OSF	Operate on Second Packet When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.
28:30 TCW	Transmit Channel Weight This field indicates the weight assigned to the corresponding Transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels.
31 ST	Start or Stop Transmission Command When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions: – The current position in the list This is the base address of the Transmit list set by the DMA_CH0_TxDesc_List_Address register. – The position at which the transmission was previously stopped If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the DMA_CH0_Status register is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the DMA_CH0_TxDesc_List_Address register, the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program DMA_CH0_TxDesc_List_Address register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

47.2.199 DMA Channel *n* Receive Control register (DMA_CH*n*_RX_CONTROL)

The DMA Channel *n* Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

Offset 0x1108 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RPF	0	0	0	0	0	0	0	0	0	RxPBL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	RBSZ														SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 799. DMA Channel *n* Receive Control register (DMA_CH*n*_RX_CONTROL)

Table 862. DMA_CH*n*_RX_CONTROL field descriptions

Field	Description
0 RPF	DMA Rx Channel0 Packet Flush When this bit is set to 1, the DMA will automatically flush the packet from the Rx Queues destined to DMA Rx Channel 0 when the DMA Rx Channel 0 is stopped after a system bus error has occurred. The flushing happens on the Read side of the Rx Queue. When this bit is set to 0 the EQOS will not flush the packet in the Rx Queue destined to DMA Rx Channel 0 after the DMA is stopped due to a system bus error.
10:15 RxPBL	Receive Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: – Set the PBLx8 mode in the DMA_CH0_Control register. – Set the PBL.

Table 862. DMA_CH*n*_RX_CONTROL field descriptions (continued)

Field	Description
17:30 RBSZ	Receive Buffer size This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16K bytes. **Note:** The buffer size must be a multiple of 4, 8, or 16 depending on the bus widths (32, 64, or 128 respectively). This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 4, 8, or 16, it may result into undefined behavior. The 3 LSB bits for 64-bit bus width are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).
31 SR	Start or Stop Receive When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets. The DMA tries to acquire descriptor from either of the following positions: The current position in the list This is the address set by the DMA_CH0_RxDesc_List_Address register. The position at which the Rx process was previously stopped If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the DMA_CH0_Status register is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the DMA_CH0_RxDesc_List_Address register, the DMA behavior is unpredictable. When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

47.2.200 DMA Channel *n* Transmit Descriptor List Address register (DMA_CH*n*_TXDESC_LIST_ADDRESS)

The Channel *n* Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Dword aligned (for 64-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to this register only when the Tx DMA has stopped, that is, the ST bit is set to zero in DMA_CH0_Tx_Control register. When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly-programed descriptor base address. If this register is not changed when the ST bit is set to 0, the DMA takes the descriptor address where it was stopped earlier.

Offset 0x1114 + *n**0x80 (*n*=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TDESLA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDESLA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 800. DMA Channel *n* Transmit Descriptor List Address register (DMA_CH*n*_TXDESC_LIST_ADDRESS)

Table 863. DMA_CH*n*_TXDESC_LIST_ADDRESS field descriptions

Field	Description
0:31 TDESLA	Start of Transmit List This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the 3 LSB bits for 64-bit data bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

47.2.201 DMA Channel *n* Receive Descriptor List Address register (DMA_CH*n*_RXDESC_LIST_ADDRESS)

The Channel*n* Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be Dword aligned (for 64-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in DMA_CH0_Rx_Control register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programed descriptor base address.

Offset 0x111C + *n**0x80 (*n*=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	RDESLA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	RDESLA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 801. DMA Channel *n* Receive Descriptor List Address register (DMA_CH*n*_RXDESC_LIST_ADDRESS)

Table 864. DMA_CH*n*_RXDESC_LIST_ADDRESS field descriptions

Field	Description
0:31 RDESLA	Start of Receive List This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the 3 LSB bits for 64-bit data bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

47.2.202 DMA Channel *n* Transmit Descriptor Tail Pointer register (DMA_CH*n*_TXDESC_TAIL_POINTER)

The Channel*n* Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

Offset 0x1120 + *n**0x80 (*n*=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TDTP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TDTP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 802. DMA Channel *n* Transmit Descriptor Tail Pointer register (DMA_CH*n*_TXDESC_TAIL_POINTER)

Table 865. DMA_CH*n*_TXDESC_TAIL_POINTER field descriptions

Field	Description
0:31 TDTP	Transmit Descriptor Tail Pointer This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

47.2.203 DMA Channel *n* Receive Descriptor Tail Pointer (DMA_CH*n*_RXDESC_TAIL_POINTER)

The Channel*n* Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

Offset 0x1128 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	RDTP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	RDTP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 803. DMA Channel *n* Receive Descriptor Tail Pointer (DMA_CH*n*_RXDESC_TAIL_POINTER)

Table 866. DMA_CH*n*_RXDESC_TAIL_POINTER field descriptions

Field	Description
0:31 RDTP	Receive Descriptor Tail Pointer This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

47.2.204 DMA Channel *n* Transmit Descriptor Ring Length register (DMA_CH*n*_TXDESC_RING_LENGTH)

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

Offset 0x112C + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	TDRL									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 804. DMA Channel *n* Transmit Descriptor Ring Length register (DMA_CH*n*_TXDESC_RING_LENGTH)

Table 867. DMA_CH*n*_TXDESC_RING_LENGTH field descriptions

Field	Description
22:31 TDRL	Transmit Descriptor Ring Length This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. It’s recommended to have a minimum ring descriptor length of 4.

47.2.205 DMA Channel *n* Receive Descriptor Ring Length register (DMA_CH*n*_RXDESC_RING_LENGTH)

The Channel*n* Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

Offset 0x1130 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	RDRL									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 805. DMA Channel *n* Receive Descriptor Ring Length register (DMA_CH*n*_RXDESC_RING_LENGTH)

Table 868. DMA_CH*n*_RXDESC_RING_LENGTH field description

Field	Description
22:31 RDRL	Receive Descriptor Ring Length This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors.

47.2.206 DMA Channel *n* Interrupt Enable register (DMA_CH*n*_INTERRUPT_ENABLE)

The Channel*n* Interrupt Enable register enables the interrupts reported by the Status register.

Offset 0x1134 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	0	0	0	TBUE	TXSE	TIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 806. DMA Channel *n* Interrupt Enable register (DMA_CH*n*_INTERRUPT_ENABLE)

Table 869. DMA_CH*n*_INTERRUPT_ENABLE field descriptions

Field	Description
16 NIE	Normal Interrupt Summary Enable When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: – Bit 31: Transmit Interrupt – Bit 29: Transmit Buffer Unavailable – Bit 25: Receive Interrupt – Bit 20: Early Receive Interrupt When this bit is reset, the normal interrupt summary is disabled.
17 AIE	Abnormal Interrupt Summary Enable When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: – Bit 30: Transmit Process Stopped – Bit 24: Rx Buffer Unavailable – Bit 23: Receive Process Stopped – Bit 22: Receive Watchdog Timeout – Bit 21: Early Transmit Interrupt – Bit 19: Fatal Bus Error When this bit is reset, the abnormal interrupt summary is disabled.
18 CDEE	Context Descriptor Error Enable When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.
19 FBEE	Fatal Bus Error Enable When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.
20 ERIE	Early Receive Interrupt Enable When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled.
21 ETIE	Early Transmit Interrupt Enable When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.
22 RWTE	Receive Watchdog Timeout Enable When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled.
23 RSE	Receive Stopped Enable When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.
24 RBUE	Receive Buffer Unavailable Enable When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.
25 RIE	Receive Interrupt Enable When this bit is set along with the NIE bit, the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.

Table 869. DMA_CH*n*_INTERRUPT_ENABLE field descriptions (continued)

Field	Description
29 TBUE	Transmit Buffer Unavailable Enable When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.
30 TXSE	Transmit Stopped Enable When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.
31 TIE	Transmit Interrupt Enable When this bit is set along with the NIE bit, the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.

47.2.207 DMA Channel *n* Receive Interrupt Watchdog Timer register (DMA_CH*n*_Rx_INTERRUPT_WATCHDOG_TIMER)

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the DMA_CH*n*_Status register.

Offset 0x1138 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RWTU	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RWT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 807. DMA Channel *n* Receive Interrupt Watchdog Timer register (DMA_CH*n*_Rx_INTERRUPT_WATCHDOG_TIMER)

Table 870. DMA_CH*n*_Rx_INTERRUPT_WATCHDOG_TIMER field descriptions

Field	Description
14:15 RWTU	Receive Interrupt Watchdog Timer Count Units This fields indicates the number of system clock cycles corresponding to one unit in RWT field. 00 256 01 512 10 1024 11 2048 For example, when RWT=2 and RWTU=1, the watchdog timer is set for 2*512=1024 system clock cycles.
24:31 RWT	Receive Interrupt Watchdog Timer Count This field indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set. The watchdog timer is triggered with the programed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the DMA_CH0_Status register, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[1]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[1] of any received packet.

47.2.208 DMA Channel *n* Slot Function Control Status register (DMA_CH*n*_SLOT_FUNCTION_CONTROL_STATUS)

The Slot Function Control and Status register contains the control bits for slot function and the status for Transmit path.

Offset 0x113C + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	RSN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ASC	ESC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 808. DMA Channel *n* Slot Function Control Status register (DMA_CH*n*_SLOT_FUNCTION_CONTROL_STATUS)

Table 871. DMA_CH*n*_SLOT_FUNCTION_CONTROL_STATUS field descriptions

Field	Description
12:15 RSN	Reference Slot Number This field gives the current value of the reference slot number in the DMA. It is used for slot comparison.
30 ASC	Advance Slot Check When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programed in the Tx descriptor is – equal to the reference slot number given in the RSN field or – ahead of the reference slot number by up to two slots This bit is applicable only when the ESC bit is set.
31 ESC	Enable Slot Comparison When set, this bit enables the checking of the slot numbers programed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is – equal to the reference slot number or – ahead of the reference slot number by one slot When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed.

47.2.209 DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_TXDESC)

The Channel*n* Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

Offset 0x1144 + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CURTDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CURTDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 809. DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_TXDESC)

Table 872. DMA_CHn_CURRENT_APP_TXDESC field descriptions

Field	Description
0:31 CURTDESAPTR	Application Transmit Descriptor Address Pointer The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

47.2.210 DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_RXDESC)

The Channel*n* Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

Offset 0x114C + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CURRDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CURRDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 810. DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_RXDESC)

Table 873. DMA_CH*n*_CURRENT_APP_RXDESC field descriptions

Field	Description
0:31 CURRDESAPTR	Application Receive Descriptor Address Pointer The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

47.2.211 DMA Channel *n* Current Application Transmit Buffer register (DMA_CH*n*_CURRENT_APP_TXBUFFER)

The Channel*n* Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

Offset 0x1154 + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CURTBUFAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CURTBUFAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 811. DMA Channel *n* Current Application Transmit Buffer register (DMA_CH*n*_CURRENT_APP_TXBUFFER)

Table 874. DMA_CH*n*_CURRENT_APP_TXBUFFER field descriptions

Field	Description
0:31 CURTBUFAPTR	Application Transmit Buffer Address Pointer The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

47.2.212 DMA Channel *n* Current Application Receive Buffer register (DMA_CH*n*_CURRENT_APP_RXBUFFER)

The Channel 0 Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

Offset 0x115C + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CURRBUFAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CURRBUFAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 812. DMA Channel *n* Current Application Receive Buffer register (DMA_CH*n*_CURRENT_APP_RXBUFFER)

Table 875. DMA_CH*n*_CURRENT_APP_RXBUFFER field descriptions

Field	Description
0:31 CURRBUFAPTR	Application Receive Buffer Address Pointer The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

47.2.213 DMA Channel *n* Status register (DMA_CH*n*_STATUS)

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

Offset 0x1160 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	REB			TEB		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	0	0	0	TBU	TPS	TI
W	NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI				TBU	TPS	TI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 813. DMA Channel *n* Status register (DMA_CH*n*_STATUS)

Table 876. DMA_CH*n*_STATUS field descriptions

Field	Description
10:12 REB	Rx DMA Error Bits This field indicates the type of error that caused a Bus Error. For example, error response on the AHB interface. Bit 10 1'b1: Error during data transfer by Rx DMA 1'b0: No Error during data transfer by Rx DMA Bit 11 1'b1: Error during descriptor access 1'b0: Error during data buffer access Bit 12 1'b1: Error during read transfer 1'b0: Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt.
13:15 TEB	Tx DMA Error Bits This field indicates the type of error that caused a Bus Error. For example, error response on the AHB interface. Bit 13 1'b1: Error during data transfer by Tx DMA 1'b0: No Error during data transfer by Tx DMA Bit 14 1'b1: Error during descriptor access 1'b0: Error during data buffer access Bit 15 1'b1: Error during read transfer 1'b0: Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Table 876. DMA_CH*n*_STATUS field descriptions (continued)

Field	Description
16 NIS	Normal Interrupt Summary Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: Bit 31: Transmit Interrupt Bit 29: Transmit Buffer Unavailable Bit 25: Receive Interrupt Bit 20: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in DMA_CH0_Interrupt_Enable register) affect the Normal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
17 AIS	Abnormal Interrupt Summary Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: Bit 30: Transmit Process Stopped Bit 24: Receive Buffer Unavailable Bit 23: Receive Process Stopped Bit 21: Early Transmit Interrupt Bit 19: Fatal Bus Error Bit 18: Context Descriptor Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
18 CDE	Context Descriptor Error This bit indicates that the DMA Tx engine received a context descriptor in the middle of a packet (in an intermediate descriptor), and the DMA Tx engine ignored it. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
19 FBE	Fatal Bus Error This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
20 ERI	Early Receive Interrupt This bit indicates that the DMA filled the first data buffer of the packet. The RI bit of this register automatically clears this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

Table 876. DMA_CH*n*_STATUS field descriptions (continued)

Field	Description
21 ETI	Early Transmit Interrupt This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
22 RWT	Receive Watchdog Timeout This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.
23 RPS	Receive Process Stopped This bit is asserted when the Rx process enters the Stopped state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
24 RBU	Receive Buffer Unavailable This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
25 RI	Receive Interrupt This bit indicates that the packet reception is complete. When packet reception is complete, Bit 0 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. The reception remains in the Running state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
29 TBU	Transmit Buffer Unavailable This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPS0 field of the DMA_Debug_Status0 register explains the Transmit Process state transitions. To resume processing the Transmit descriptors, the application should do the following: 1. Change the ownership of the descriptor by setting Bit 0 of TDES0. 2. Issue a Transmit Poll Demand command. For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

Table 876. DMA_CH*n*_STATUS field descriptions (continued)

Field	Description
30 TPS	Transmit Process Stopped This bit is set when the transmission is stopped. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
31 TI	Transmit Interrupt This bit indicates that the packet transmission is complete. When transmission is complete, Bit 0 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

47.2.214 DMA Channel *n* Miss Frame Counter register (DMA_CH*n*_MISS_FRAME_CNT)

This register has the number of packet counter that got dropped by the DMA either due to Bus Error or due to programing RPF field in DMA_CH0_Rx_Control register.

Offset 0x116C + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MFCO	0	0	0	0	MFC										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 814. DMA Channel *n* Miss Frame Counter register (DMA_CH*n*_MISS_FRAME_CNT)

Table 877. DMA_CH*n*_MISS_FRAME_CNT field descriptions

Field	Description
16 MFCO	Overflow status of the MFC Counter When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 MFC	This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in DMA_CH0_Rx_Control register. The counter gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.

47.3 Functional Description

47.3.1 AHB Master Interface

The DMA controller interfaces with the application through the AMBA AHB interface. The AHB master interface converts the internal DMA request cycles into AHB cycles. The AHB master interface has the following features:

- The AHB master interface is AMBA 2.0-compliant AHB master with no restrictions
- The AHB master interface supports the following burst length modes: fixed, unspecified, and mixed. These modes are explained in *Section 47.3.1.1: AHB Burst Length Modes*
- The AHB master interface can handle the AHB SPLIT, RETRY, and ERROR conditions. Any ERROR response halts all further transactions for that DMA and indicates the error as fatal through CSR and interrupt. The application can choose to give a hard or soft reset to the module to restart the operation
- The AHB master interface can handle the AHB 1 K boundary breaking
- The AHB master interface can handle all 64-bit data transfers except for Descriptor Status Write accesses (which may be 32-bit). In any burst data transfer, the address bus value is always aligned to the data bus width and need not be aligned to the beat size
- The AHB master interface can align all AHB burst transfers to an address value. For more information, refer to *Section 47.3.1.2: Aligning AHB Burst Transfers to an Address Value*
- The DMA requests an AHB Burst Read transfer only when it can completely accept the received burst data. The data read from the AHB is always pushed into the DMA without any delay or Busy cycles. However, when multiple channels are selected, there may be some delay and busy cycles may be inserted because of the shared-memory architecture for the channels
- The DMA requests an AHB Burst Write transfer only when it has the sufficient data to completely transfer the burst. The AHB interface always assumes that it has the data available to push into the AHB bus. However, the DMA can prematurely indicate end-of-valid data (because of the transfer of EOP of an Ethernet packet) during the burst
- In Fixed Burst Length mode, the AHB master interface continues the burst with dummy data until the specified length is completed. In INCR mode, it ends the burst transfer prematurely

47.3.1.1 AHB Burst Length Modes

The AHB master interface supports the following burst length modes:

- Fixed Burst Length: In fixed burst length mode, the AHB master always initiates a burst with SINGLE, INCR4, INCR8, or INCR16 type. However, when such a burst is responded with SPLIT, RETRY, or Early Burst Termination (EBT), by default, the AHB master again initiates the pending transfers of the burst with INCR or SINGLE burst length type. When only one beat is remaining, the AHB master performs a SINGLE transfer. If more than one beat is remaining, the AHB master performs an INCR transfer. The AHB master terminates such INCR bursts when remaining beats of the original requested fixed burst are transferred.
  To ensure that the AHB master always performs the INCRx and SINGLE transfers even when a part of the burst is initiated again after a SPLIT, RETRY, or EBT response, use the RIB bit of the register. However, when you enable this mode, the pending burst may

further get split into multiple transfers as per the following rules before any new transfer is initiated:

   – If the DMA requests a burst transfer that is not equal to INCR4, INCR8, or INCR16, the AHB application interface splits the transfer into multiple burst transactions. For example, if the DMA requests a 15-beat burst transfer, the AHB interface splits it into multiple transfers of INCR8 and INCR4, and three SINGLE transactions
   – If the burst length is more than 16, the AHB master interface splits it into multiple burst transfers. For example, if the burst length is 32 and the DMA starts a 32-beat transfer, the AHB master splits the burst into two INCR16

- Unspecified Burst Length: In unspecified burst mode, the AHB master always initiates a transfer with INCR and completes the DMA requested burst in one transfer
- Mixed Burst Mode: In mixed burst mode, the AHB master always initiates the bursts with fixed-size (INCRx) when the DMA requests transfers of size less than or equal to 16 beats. When the DMA requests bursts of length more than 16, the AHB master initiates such transfers with INCR and completes the request in one transfer

To ensure that the pending burst of a transfer is rebuilt only with INCRx transfers when an INCRx transfer is interrupted by RETRY, SPLIT, or EBT response, use the RB bit of the DMA_SysBus_Mode register. However, if an INCR burst initiated in this mode is interrupted, the AHB master rebuilds the burst only with INCR transfers.

You can choose the burst modes by programming the FB bit in the DMA_SysBus_Mode register.

Note: The AHB master does not generate the WRAP bursts.

The AHB master does not generate the locked or protected transfers. Therefore, the Ethernet IP does not have the HLOCK or HPROT output signals.

47.3.1.2 Aligning AHB Burst Transfers to an Address Value

The AHB master interface can align all AHB burst transfers to an address value. You can enable this by using the AAL bit in the DMA_SysBus_Mode register. If Bit 31 and Bit 19 of DMA_SysBus_Mode register are set to 1, the AHB interface and the DMA ensure that all initiated beats are aligned to the address, completing the packet transfer in the minimum number of required beats.

The AHB interface and the DMA split unaligned address beats for a 64-bit data bus and initiate INCR16 beats only when the seven least-significant bits of the address have a zero value.

47.3.2 DMA Controller

The DMA has independent Transmit (Tx) and Receive (Rx) engines, and a CSR space. The Tx engine transfers data from the system memory to the device port (MTL), whereas the Rx engine transfers data from the device port to the system memory. The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx Buffer in system memory and Tx data packets from the Tx Buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The DMA supports up to 2 Tx and 2 Rx Descriptor lists (or DMA channels). The base address of each list is written to the respective Tx Descriptor List Address register and Rx Descriptor List Address register. The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. You can control the offset by the DSL field of DMA_CH*n*_Control register. The number of descriptors in the list is programed in the respective Tx (or Rx) Descriptor Ring Length register. Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List Address register to create a descriptor ring.

The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet but cannot exceed a single packet. Buffers contain only data. Buffer status is maintained in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

The Ethernet IP supports the ring structure for the DMA descriptor. For more information on the descriptor, refer to *Section 47.3: Functional Description* that describes the descriptor structure and how the DMA accesses the descriptors.

47.3.2.1 Application Bus Burst Access

When configured through Bit 31 of the DMA_SysBus_Mode register, the DMA attempts to execute the fixed-length burst transfers on the AHB master interface. The PBL field of Transmit Control and Receive Control registers of respective DMA channel indicates and limits the maximum burst length. The Rx and Tx descriptors are always accessed in the maximum possible (limited by PBL or 16 * 8/bus width) burst length for 16 bytes to be read.

The Tx DMA initiates a data transfer only when sufficient space is available in the MTL Tx Queue to accommodate either of the following:

- Bytes corresponding to the configured burst (PBL * bus_width/8)
- Remaining bytes in Tx Buffer without EOP
- Number of bytes till EOP

The Rx DMA initiates a data transfer in the following conditions:

- Sufficient data is available in MTL Rx Queue to accommodate the configured burst
- EOP (when it is less than the configured burst length) is detected in the Rx Queue

The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB Interface is configured for fixed-length burst, it transfers the data by using the best combination of INCR4, INCR8, or INCR16 and SINGLE transactions. If EOP is reached before the fixed-burst ends on the AHB interface, dummy transfers are performed in-order to complete the fixed-burst. Otherwise, (Bit 31 of DMA_SysBus_Mode register is reset), the DMA transfers the data using INCR (undefined length) and SINGLE transactions.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer initiated by the AHB is less than or equal to the size of the configured PBL. Therefore, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for PBL > 16) for AHB interface because it does not support more than INCR16.

47.3.2.1.1 Application Data Buffer Alignment

The Tx and Rx data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for buffers can be aligned to any of the four bytes. However, the DMA always initiates write transfers with address aligned to the bus width and dummy data (old data) in the invalid byte lanes. This typically happens during the transfer of the beginning or end of an Ethernet packet. The software driver should discard the dummy bytes based on the start address of the buffer and size of the packet.

Table 878. Application Data Buffer Alignment Examples

Data Buffer	Examples
Buffer Read	If the Tx buffer address is 32’h00000FF2 (for 32-bit data bus), and 15 bytes need to be transferred, the DMA reads five full words from address 32’h00000FF0, but when transferring data to the MTL Tx queue, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures that it transfers a full 32-bit data to the MTL Tx queue, unless it is the end of packet.
Buffer Write	If the Rx buffer address is 32’h0000FF2 (for 64-bit data bus) and 16 bytes of a received packet need to be transferred, the DMA writes 3 full words from address 32’h00000FF0. However, the first 2 bytes of the first transfer and the last 6 bytes of the third transfer have dummy data. The DMA considers the offset address only if it is the first Rx buffer of the packet. The DMA ignores the offset address and performs full word writes for the middle and the last Rx buffer of the packet.

47.3.2.1.2 Buffer Size Calculations

The DMA does not update the size fields in the Tx and Rx descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The Tx DMA transfers the exact number of bytes (indicated by buffer size field of TDES2) towards the MAC. If a descriptor is marked as first (FS bit of TDES2 is set), the DMA marks the first transfer from the buffer as SOP. If a descriptor is marked as last (LS bit of TDES2), the DMA marks the last transfer from that data buffer as EOP to the MTL.

The Rx DMA transfers data to a buffer until the buffer is full or the end of packet is received from the MTL. When the FS bit of a descriptor is set, the amount of valid data in a buffer is accurately indicated by the buffer size field (programed in DMA Channel Receive Control register) minus the data buffer pointer offset. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, the buffer may not be full (as indicated by the buffer size in Bits[17:30] of Receive Control register). To compute the amount of valid data in this final buffer, the driver must read the packet length (FL bits of RDES3[17:31]) and subtract the sum of the buffer sizes of the preceding buffers in this packet. The Rx DMA always transfers the start of next packet with a new descriptor.

Note: Even when the start address of a Rx buffer is not aligned to the data width of system bus, the system should allocate a Rx buffer of a size aligned to the system bus width. For example, if the system allocates a Rx buffer of 1024 bytes (1 KB) starting from address 0x1000, the software can program the buffer start address in the Rx descriptor to have a 0x1002 offset. The Rx DMA writes the packet to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual packet is written from location 0x1002. Therefore, the actual useful space in this buffer is 1022 bytes, even though the buffer size is programed as 1024 bytes, because of the start address offset.

47.3.2.1.3 DMA Arbiter

The arbiter inside the DMA module performs the arbitration between the Tx and Rx channel accesses to the AHB master interface. The following two types of arbitrations are supported:

- Round-Robin Arbitration: When Bit 30 of the DMA_Mode register is reset and both Tx and Rx DMAs are simultaneously requesting for access, the arbiter allocates the data bus in ratio set by Bits[17:19] of DMA_Mode register
- Fixed-Priority Arbitration: When Bit 30 of the DMA_Mode register is set, the Rx DMA always gets priority over the Tx DMA for data access by default. When Bit 20 of DMA_Mode register is also set, the Tx DMA gets priority over the Rx DMA

47.3.2.2 DMA Transmission

47.3.2.2.1 TX DMA Operation: Default (Non-OSP) Mode

The TX DMA engine in default mode proceeds as follows:

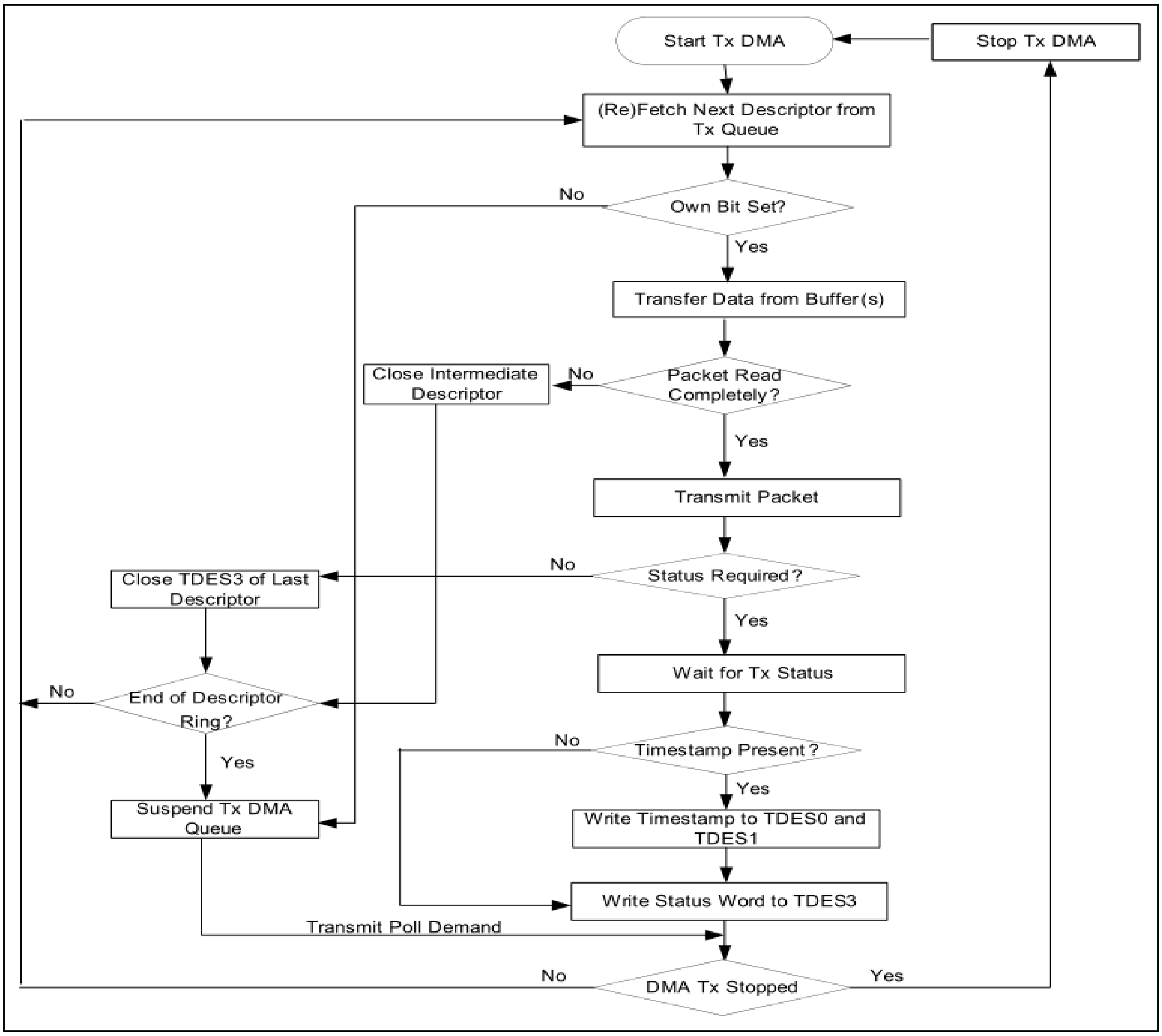
1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[0]) after setting up the corresponding data buffer(s) with Ethernet Packet data
2. The application advances the Descriptor Tail pointer offset value of the Transmit Channel
3. While in the Run state, the DMA runs an Arbitration cycle to select the next Tx DMA channel from which the packets requiring transmission should be processed
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended and Bit 29 and Bit 15 of Status Register of corresponding DMA channel are set and the Tx Engine proceeds to Step *11*:
   a) The descriptor is flagged as owned by the application (TDES3 [0] = 1'b0)
   b) The Descriptor Tail pointer is equal to the Current Descriptor pointer in Ring Descriptor list Mode
   c) An error condition occurs
6. If the acquired descriptor is flagged as owned by the DMA (TDES3[0] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor
7. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission
8. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Step *3* through Step *7* are repeated until the end-of-Ethernet-packet data is transferred to the MTL
9. When packet transmission is complete, if IEEE 1588 Timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The

status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step.
If timestamp feature is not enabled for this packet, the DMA does not alter the contents of TDES0 and TDES1

10. Bit 31 of Status Register of corresponding channel is set after completing transmission of a packet that has Interrupt on Completion (TDES2[0]) set in its Last Descriptor. The DMA engine returns to Step 3
11. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to Step 3). A poll demand command is triggered by writing any value to the DMA_CH0_TxDesc_Tail_Pointer when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 31 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state

The Tx DMA transmission flow in default mode is shown in Figure 815.

Figure 815. TX DMA Operation in Default Mode



47.3.2.2.2 TX DMA Operation: OSP Mode

In the Run state, if Bit 27 is set in the Transmit Control Register of corresponding DMA channel, the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. As the Transmit process finishes transferring the first packet, it immediately polls the Transmit Descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

In OSP mode, the Run state Tx DMA operates in the following sequence:

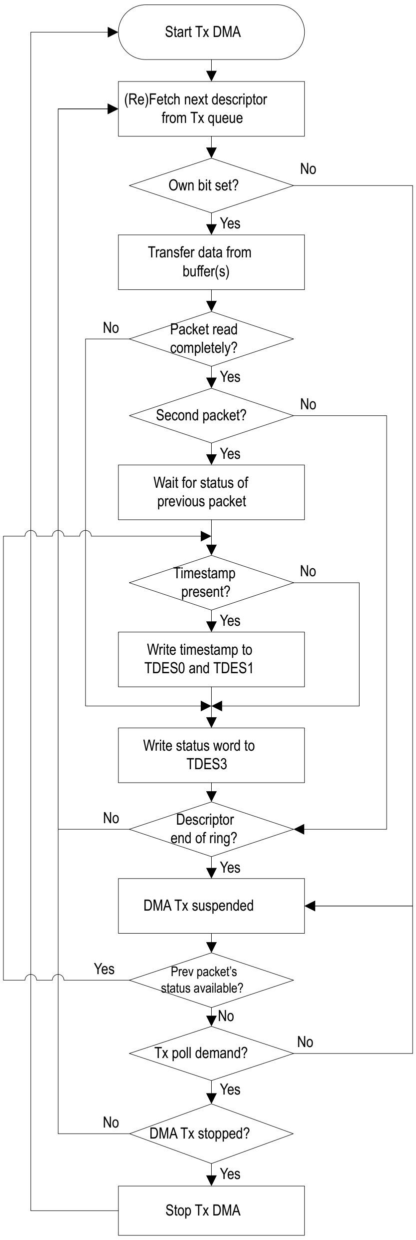
1. The DMA operates as described in Step *1* – Step *7* of the Tx DMA (default mode)
2. The DMA fetches the next descriptor without closing the last descriptor of previous packet
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and skips to Step *7*
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor.
   If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES0 and TDES1
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to Step *3* (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (Step *7*)
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA does the following:
   a) Writes the timestamp (if enabled for the current packet) to TDES0 and TDES1
   b) Writes the status to the corresponding TDES3
   c) Sets relevant interrupts and returns to Suspend mode

   If no status is pending and the application stopped the DMA by clearing Bit 31 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state
8. The DMA can exit Suspend mode and enter the Run state (goes to Step *1* or Step *2* depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel

Note: The DMA fetches the next descriptor before closing the current descriptor. Therefore, the descriptor ring length must be more than two. It is recommended to have a minimum descriptor length of four.

The basic flow is described in *Figure 816*.

Figure 816. Tx DMA Operation in OSP Mode



47.3.2.2.3 Transmit Packet Processing

The Tx DMA expects that the data buffers contain complete Ethernet packets, excluding preamble, pad bytes, and FCS fields. The DA, SA, and Type/Length fields contain valid data. If the Tx Descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet packets (excluding preamble), including the CRC bytes.

Packets can be data-chained and can span several buffers. Packets must be delimited by the First Descriptor (TDES3[2]) and the Last Descriptor (TDES3[3]). As transmission starts, the First Descriptor must have TDES3[2] set. When this occurs, the packet data is transferred from the application buffer to the MTL Tx Queue. Concurrently, if the current packet has the Last Descriptor (TDES3[3]) clear, the Tx Process attempts to acquire the Next Descriptor. The Tx Process expects this descriptor to have TDES3[2] clear. If TDES3[3] is clear, it indicates an intermediary buffer. If TDES3[3] is set, it indicates the last buffer of the packet.

After the last buffer of the packet has been transmitted, the DMA writes back the final status information to the Transmit Descriptor 3 (TDES3) word of the descriptor that has the Last Descriptor Bit set in Transmit Descriptor 3 (TDES3[3]). At this time, if Interrupt on Completion (TDES2[0]) is set, Bit 31 of Status Register of corresponding DMA channel is set, the Next Descriptor is fetched, and the process repeats. The actual packet transmission begins after either of the following:

- The MTL Tx Queue has reached a programmable Transmit threshold (Bits[25:27] of Transmit Operation Mode register of corresponding MTL Transmit Queue)
- A full packet is contained in the FIFO

You can also use the store-and-forward mode (Bit 30 of MTL Transmit Operation Mode Register of a queue). In this mode, descriptors are released (Own bit TDES0[0] clears) when the DMA finishes transferring the packet.

Note: To ensure proper transmission of a packet and the next packet, you must specify a non-zero buffer size for the Transmit descriptor that has the Last Descriptor (TDES3[3]) set.

47.3.2.2.4 Transmit Polling Suspended

Transmit polling can be suspended by any of the following conditions:

- The DMA detects a descriptor owned by the application (TDES3[0]=0).
  To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command by writing the Tail Pointer register. If the DMA goes into SUSPEND state because of this condition, Bit 16 and Bit 29 of Status Register of corresponding DMA channel are set
- A packet transmission is aborted when a Transmit error is detected because of underflow.
  The appropriate Transmit Descriptor 3 (TDES3) bit is set. When this condition occurs, the following bits are set and the information is written to Transmit Descriptor 0, causing the suspension:
  - Bit 17 of Status Register of corresponding DMA channel
  - Transmit Underflow bit of corresponding queue in MTL_Interrupt_Status
- The DMA detects that the Tail Pointer is equal to the Current descriptor closed by it.
  To resume, the software driver must modify the Tail Pointer register

In all conditions, the position in the Transmit List is retained. The retained position is that of the descriptor following the Last Descriptor closed by the DMA. The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension cause.

47.3.2.2.5 Transmit Channel Arbitration

An arbiter provides access to multiple DMAs trying to access the Bus Interface Unit (BIU). *Figure 817* shows the DMA Tx Channel Arbitration Process.

When there is any request in the Tx queue, the DMA arbiter checks the type of the request: packet buffer fetch or descriptor fetch request. The descriptor fetch requests have higher priority than the buffer requests. Therefore, when there is a descriptor fetch request, the

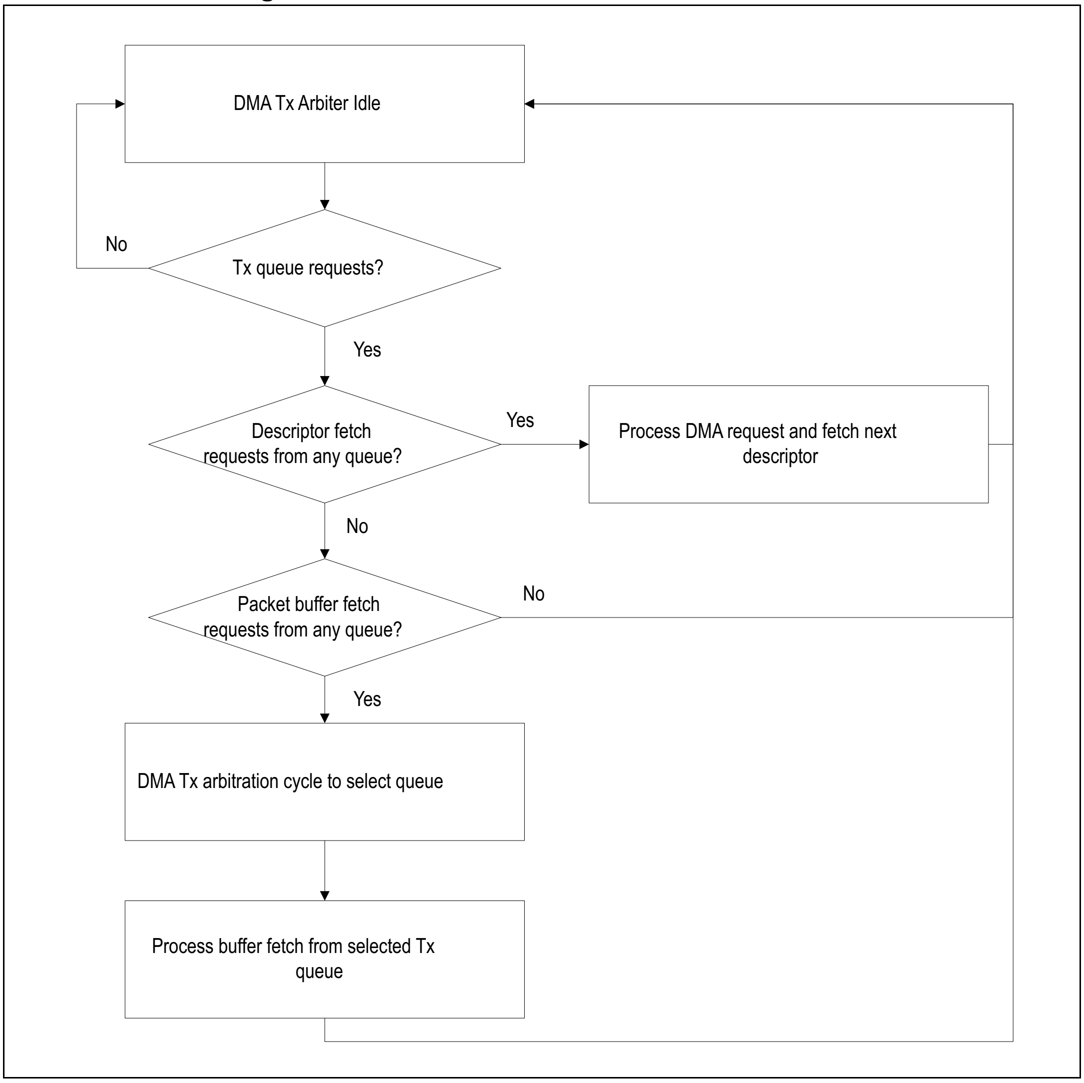
DMA arbiter acknowledges the DMA channel that is requesting for a descriptor fetch. If there is no descriptor fetch request, the arbiter looks for packet buffer fetch requests.

The DMA arbiter acknowledges the descriptor fetch request of one DMA channel at a time. Descriptor fetch requests are granted using a fixed priority with the higher channel having higher priority (Channel 1 having priority over Channel 0). For packet buffer fetches, the DMA arbiter uses the programed channel weight and priority to decide which channel to acknowledge. The DMA arbiter performs a burst-by-burst arbitration based on one of the following algorithms:

- Weighted Strict Priority (WSP): In WSP arbitration mode, the arbiter first processes Channel 1 (or the last enabled channel) and then Channel 0. If any channel does not have a frame to transmit, the weight of that channel gets reassigned to Channel1 (or last enabled channel). If Channel 1 has no frames to transmit, the remaining weight is assigned to Channel 0
- Weighted Round Robin (WRR): In WRR arbitration mode, the arbiter first selects the channel with the highest weight programed, and then the channel with next highest weight, and so on. If any channel does not have a frame to transmit, the weight of that channel gets equally distributed to all channels that have frames to transmit
- Fixed priority (FP): In Fixed priority mode, Channel 0 has the lowest priority and the last selected channel has the highest priority. The weight programed in the Transmit Control register of a channel is ignored

In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

Figure 817. DMA Tx Channel Arbitration Process



47.3.2.3 DMA Reception

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel. The ARI data at the start of the frame indicates the channel number to which the current frame must be written. If only one DMA channel is selected, this information is not provided.

Figure 818 shows the reception sequence for Rx DMA engine. The following list describes this sequence:

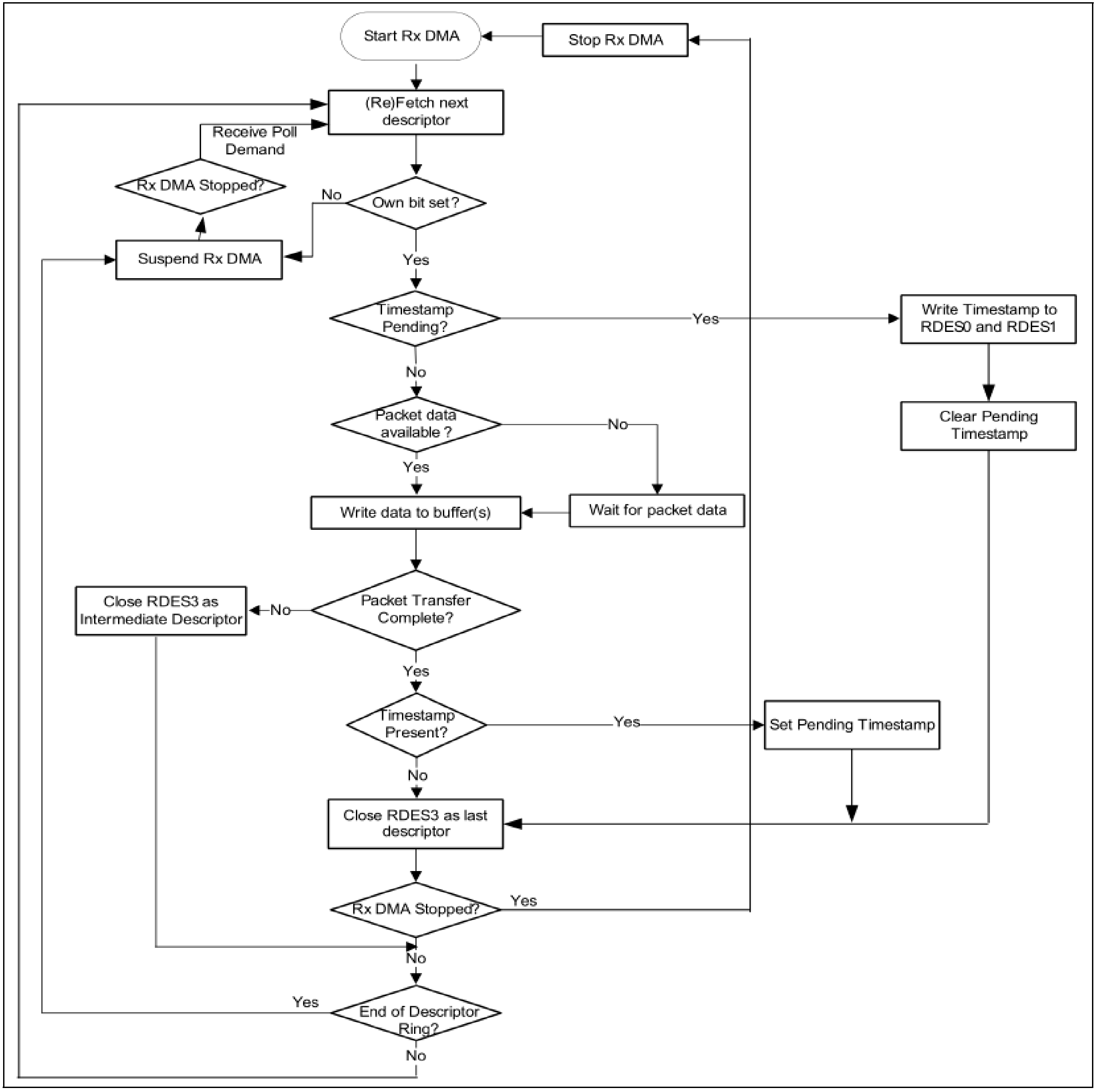
1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[0]).
   The application must set the correct value in the Receive Descriptor Tail Pointer register of corresponding DMA channel
2. When Bit 31 of Receive Control register of corresponding DMA channel is set, the DMA enters the Run state.
   The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor

Tail Pointer register values. If there are no free descriptors, the DMA Channel enters the suspend state and goes to Step *11*

3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the RDES3[CTXT] field
5. The DMA processes the incoming packets and places these in the data buffers of acquired descriptor
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to Step *10*
7. The DMA takes the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last Descriptor bit set
8. The DMA writes the Frame Length to RDES3 and VLAN Tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor
9. If IEEE 1588 Timestamp feature is enabled, the DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor)
10. If more descriptors are available in the Rx DMA Descriptor Ring, go to step *3*; otherwise, go to the Suspend state (Step *11*)
11. The Receive DMA exits the Suspend state when a Receive Poll demand is given and the application advances the Receive Tail Pointer register of a channel

The engine proceeds to Step *2* and re-fetches the next descriptor.

Figure 818. RX DMA Operation



47.3.2.3.1 Receive Descriptor Acquisition

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming packet. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- Bit 31 of Receive Control Register of corresponding DMA channel is set immediately after being placed in the Run state
- The Descriptor Tail pointer register value is ahead of the Current Descriptor acquired by the Rx DMA
- The controller has completed packet reception, but the current Receive Descriptor is not yet closed
- A Receive poll demand is issued (update of the Tail Pointer register)

47.3.2.3.2 Receive Packet Processing

The sequence for processing a Receive packet is as follows:

1. The MAC transfers the received packets to the MTL memory only if the packet passes the address filter. If the packet fails the address filtering, it is dropped in the MAC block (unless Bit 0 of MAC_Packet_Filter register is set)
2. If packet size is greater than or equal to configurable threshold bytes set for Rx Queue of MTL, or when the complete packet is written to the queue in the store-and-forward mode, the MTL block requests the DMA block to begin transferring the packet data to the Receive Buffer pointed by the current descriptor.
   Packets smaller than 64 bytes, because of collision or premature termination, are removed from the MTL Receive Queue
3. When the DMA application Interface (AHB) becomes ready, it transfers the data and sets the following:
   a) If the packet fits in a single descriptor, the DMA sets both Last Descriptor (RDES3[3]) and First Descriptor (RDES3[2])
   b) If the packet fits into more than one descriptor, the DMA sets the First Descriptor (RDES3[2]) to delimit the packet
4. The DMA releases the descriptors by resetting the Own (RDES3[0]) bit to 0, either because the Receive buffer filled up or the last segment of the packet is transferred to the Receive buffer. The received packets status is updated in the last descriptor
5. If Interrupt Enabled on Completion (RDES3[1]) bit is set in any of the Descriptors between the First and Last Descriptor of the Packet and Bit 25 of Interrupt Enable Register of corresponding DMA channel is set, the DMA sets Bit 25 of Status register of corresponding DMA channel.
   The same process repeats unless the DMA encounters a descriptor flagged as being owned by the application or when there are no more descriptors in the ring. When the DMA finds a descriptor owned by the application and if Bit 24 of Interrupt Enable Register of corresponding DMA channel is set, the Receive Process sets Bit 24 of Status register of corresponding DMA channel and then enters the Suspend state. The position in the receive list is retained

47.3.2.4 DMA Error Response

For any data transfer initiated by a DMA channel, if the slave replies with an error response, the DMA stops all operations and updates the error bits and the Fatal Bus Error bit in the Status Register of corresponding DMA channel. The application can either perform a reset to Ethernet IP or re-initialize the DMA descriptor list and start again. The rest of the DMA channels are not affected by such errors. In AHB the DMA receives the error response through hresp_i signal.

47.3.3 MAC Transaction Layer (MTL)

The MAC Transaction Layer (MTL) provides the FIFO memory Interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer has two data paths: Transmit path and Receive Path. The data path for both directions is 64-bit wide.

47.3.3.1 MTL Transmit Path

The DMA pushes the Ethernet packets read from the application or system memory into the corresponding queue. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application. The fill level of the queue is indicated to the internal DMA (using PBL and watermark) so that it can initiate a data fetch in required bursts from the application or system memory. The following two modes of operation trigger reading of the data towards the MAC:

- Threshold mode: In Threshold (or cut-through) mode, as soon as the number of bytes in the Queue cross the configured threshold level (or when the end of packet is written before the threshold is crossed), the data is ready to be popped out and forwarded to the MAC. The threshold level is configured by using the TTC bits of MTL_TxQ0_Operation_Mode register corresponding to an MTL queue
- Store-and-forward mode: In store-and-forward mode, the MTL pops out the packet towards the MAC only when one or more of the following conditions are true:
  - A complete packet is stored in the Queue
  - The Tx FIFO becomes almost full
  - The ATI watermark becomes low

The watermark becomes low when the requested Queue does not have space to accommodate the requested burst length on the ATI. Therefore, the MTL when operating in the store-and-forward mode allows packet transmission even if the packet length is bigger than the Tx Queue size.

The application can flush complete content of the Tx Queue by setting Bit 31 (FTQ) of Transmit Operation Mode register corresponding to an MTL queue. This bit is self-clearing and initializes the Queue pointers to the default state. If the FTQ bit is set during a packet transfer from the MTL to the MAC, the MTL stops further transfer because the queue is considered to be empty. Therefore, an underflow event occurs at the MAC transmitter.

47.3.3.1.1 Initialization Flow

Upon reset, the MTL is ready to manage the flow of data to and from the application or DMA, and MAC.

First you need to initialize the Queue size for each of the queues by programming the TQS bits of MTL_TxQ0_Operation_Mode register corresponding to a Transmit queue. You also need to initialize the MAC block. The internal DMA controllers must be individually enabled through their respective Cars.

47.3.3.1.2 Data Flow between the DMA and the MTL

This section describes the interaction process between the MTL and the DMA.

47.3.3.1.2.1 Single-Packet Transmit Operation

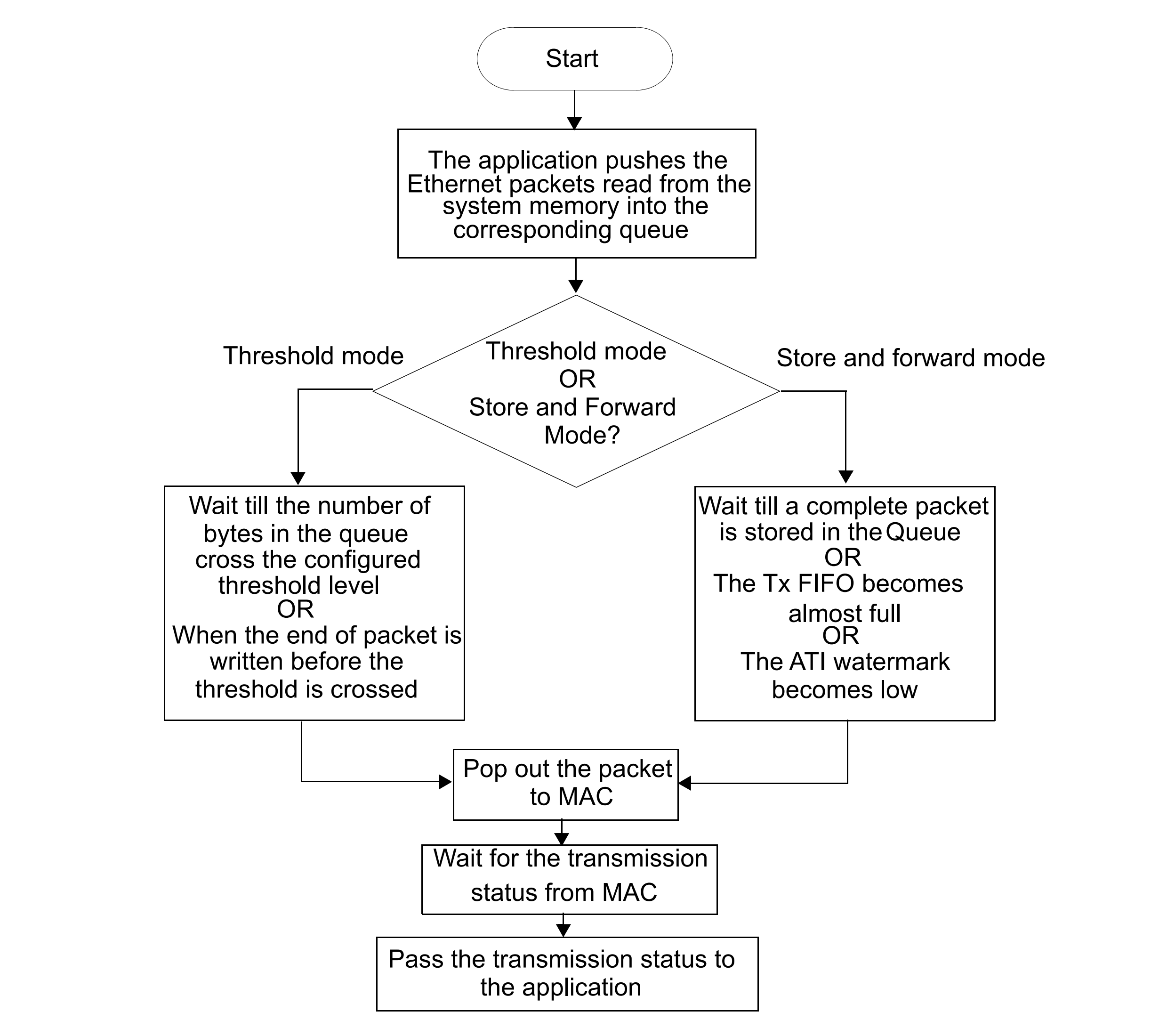
The following flow is valid when the core is operating in non-OSP mode, that is, when bit 27 (OSP) of the DMA_CH*n*_TX_Control register of the DMA channel is set to 0.

During a Transmit operation, the MTL block is a slave for the DMA controller. The general sequence of events for a Transmit operation is as follows:

1. If the system has data to be transferred, the DMA controller (if enabled) fetches the data from the application through the AHB master interface and starts forwarding it to the MTL. The DMA descriptor memory contains the information related to packet control which is used to drive the MTL Control Word. The DMA data buffer memory contains the packet data
2. The MTL pushes the data received from the DMA into the corresponding queue. This process continues until the EOP is transferred
3. When the threshold level is crossed or a full packet of data is received into the queue, the MTL reads the packet data and drives it to the MAC
4. The queue controller continues to transfer data from the queue until a complete packet is transferred to the MAC
5. When the packet transfer is complete, the MTL receives the status from the MAC and notifies the DMA controller

Figure 819 illustrates the MTL single-packet transmit operation.

Figure 819. MTL Single Packet Transmit Flow



Note: When Bit 30 (DTXSTS) of Section 47.2.163 register is set, the MTL does not provide any status to the DMA.

47.3.3.1.2.2 Transmit Operation: Two Packets in the Buffer

The following flow is valid when the core is operating in OSP mode, that is, when bit 27 (OSP) of the DMA_CH*n*_TX_Control register of the DMA channel is set to 1.

The flow is similar to the flow described in Single-Packet Transmit Operation. However after fetching a packet, the DMA instead of waiting for the status, it continues to fetch another packet if available in the system memory. So, MTL can receive the second packet while it is processing the first packet. This flow improves the performance because the DMA can process two packets back to back before waiting for the status of the first packet.

47.3.3.1.2.3 Relationship between the Number of TX DMA Channels and Tx Queues

The number of Tx DMA channels is always equal to the Tx queues. This is because the DMA is designed to arbitrate among multiple channels (to fetch the descriptor and packet data from the system memory) in terms of PBLs (Programmable Bursts Lengths). However, the Tx queues are packet-based storage memory. Therefore, one to one mapping is required between the DMA channels and Tx queues to maintain the packet-level coherency.

47.3.3.1.2.4 Retransmission during Collision

While a packet is being transferred from the MTL to the MAC, a collision event can occur on the MAC line interface in half-duplex mode. The MAC indicates a Retry attempt to the MTL by giving the status even before the EOP is transferred from the MTL. The MTL then enables the retransmission by popping out the packet again from the queue.

After more than 96 bytes are read out towards the MAC, the Queue Controller frees up that space and makes it available to the application or the DMA to push in more data. This means that the retransmission is not possible after this threshold is crossed or when the MAC indicates a late-collision event.

When a packet transmission is aborted because of underflow and a collision event immediately follows (initiating a retry), retry has higher priority than abort.

Note: Before issuing the Transmit Queue Flush command, make sure that the Tx Queue is not empty, that is, Bit 27 (TXQSTS) of Transmit Debug Register of corresponding queue is reset.

47.3.3.1.2.5 Transmit Status Word

At the end of transfer of the Ethernet packet to the MAC and after the MAC completes the transmission of the packet, the MTL provides the Transmit status on respective queues. The detailed description of the Transmit Status is the same as for Bits[17:31] of TDES3 normal descriptor in write-back format. If IEEE 1588 Timestamp feature is enabled, the MTL returns 64-bit timestamp of the packet, along with the Transmit status.

If Bit 30 (DTXSTS) of MTL_Operation_Mode register is set then it disables the dependency of the DMA to read the status words from the Status FIFO thus the DMA can push as many packets as possible into the Tx queues because the application or DMA is not required to read the status words from the Status FIFO.

47.3.3.2 Receive Path

The MTL Rx module receives the packets from the MAC and pushes them into the Rx Queue. The status (fill level) of the queue is indicated to the DMA when it crosses the configured Receive threshold (RTC bits[30:31] of MTL_RxQ0_Operation_Mode register of corresponding MTL queue), or the complete packet is received. The MTL also indicates the fill level of the queue so that the DMA can initiate pre-configured burst transfers towards the AHB interface.

47.3.3.2.1 Receive Operation

During a Receive operation, the MTL is a slave for the MAC. The general sequence of events is as follows:

1. When the MAC receives a packet, it indicates the availability of receive data
2. The MAC indicates the SOP and EOP delimiters
3. The MTL accepts the data and pushes it into corresponding Rx queue.
4. After the EOP is transferred, the MAC drives the status word which is also pushed into the corresponding Rx queue by the MTL
5. If IEEE 1588 timestamp feature is enabled and the 64-bit timestamp is available along with the packet status, it is pushed into the Rx queue as a part of the status word. For the 64-bit bus, one additional location is taken
6. The MTL takes the data out of the queue and sends it to the DMA depending on the below two modes
7. Threshold mode: In the (default) Threshold mode, the MTL reads the data and indicates its availability to DMA when one of the following happens:
   a) data bytes equal to the threshold amount are written to the Rx queue (RTC bits[30:31] of MTL_RxQ0_Operation_Mode register of corresponding MTL queue)
   b) a full packet of data is received into the queue
8. Store-and-forward mode: In the store-and-forward mode (when Bit 26 of MTL_RxQ0_Operation_Mode register of a queue is set to 1), the initial Rx queue locations are reserved for the status words before writing the SOP. A packet is read out only after it is completely written into the Rx queue. In this mode, all error packets are dropped (if configured through Bit 27 of MTL_RxQ0_Operation_Mode register of a queue) such that only valid packets are read and forwarded to the application

Note: In Threshold (cut-through) mode, the status words are stored after the packet EOP. In store-and-forward mode, the location for the maximum status words are reserved before writing the SOP and the status is written to reserved locations after writing the EOP.

47.3.3.2.2 Receive Operation Multi-Packet Handling

In Threshold mode, the packet status is available immediately after the packet data. In store-and-forward mode, the packet data is available after the packet status. The MTL is capable of storing any number of packets into the queue as long as it is not full.

If the MAC receives a packet when the corresponding Rx queue is full, the MTL ignores that packet. In addition, the MTL increments the overflow counter in the MTL_RxQ0_Missed_Packet_Overflow_Cnt of corresponding queue.

47.3.3.2.3 Error Handling

If the MTL Rx queue is full before it receives the EOP data from the MAC, the following happens:

1. An overflow is declared
2. The whole packet (including the status word) is dropped
3. The overflow counter in the DMA (Overflow Counter register of corresponding MTL queue) is incremented

This is true even if Bit 27 (FEP) of MTL_RxQ0_Operation_Mode register of corresponding MTL queue is set.

If the start address of such a packet has already been transferred to the Read Controller, the rest of the packet is dropped and a dummy EOP is written to the queue along with the status word with overflow status. The status indicates a partial packet because of overflow. In such packets, the Packet Length field is invalid. If the MTL Receive Queue is configured to operate in the store-and-forward mode and the length of the received packet is more than the queue size, overflow occurs and all such packets are dropped.

The MTL Rx Control logic can filter error and undersized packets, if enabled by using the FEP and FUP bits of MTL_RxQ0_Operation_Mode register of corresponding MTL queue. If the start address of such a packet has already been transferred to the Rx Queue Read Controller, that packet is not filtered. The start address of the packet is transferred to the Read Controller after the packet crosses the receive threshold set by Bits[30:31] of MTL_RxQ0_Operation_Mode register of corresponding MTL queue.

If the MTL Receive Queue is configured to operate in the store-and-forward mode, all error packets can be filtered and dropped. The MTL then starts the transfer of next packet (if available).

47.3.3.2.4 Receive Status Word Format

The Rx packet status words are sent to the application after the packet data in Threshold mode and before the packet data in the Store and Forward Mode.

47.3.4 MAC

The MAC supports MII and RMII interfaces towards the PHY chip. The PHY interface can be selected only once after reset.

47.3.4.1 MAC Transmission

The transmission process is as follows:

1. Transmission is initiated when the MTL pushes in data with the SOP
2. When the SOP is detected, the MAC accepts the data and begins transmitting to the MII. The time required to transmit the packet data to the MII after the application initiates the transmission depends on delay factors such as IPG delay, time to transmit preamble or SFD, and any back-off delays for half-duplex mode. While the packets data is being transmitted, the MAC can throttle the MTL
3. After the EOP is transferred to the MAC, the MAC does one of the following:
   a) The MAC completes the normal transmission and gives the transmission status to the MTL
   b) If a normal collision (in half-duplex mode) occurs during transmission, the MAC gives the Transmit Status with retry bit set to the MTL. The MAC gives the Retry request till either the packet is successfully transmitted or maximum retry requests expire
4. When maximum retry requests expire, the MAC aborts the packet transmission with Excessive Collision Transmit Status. The MAC accepts and drops all further data until

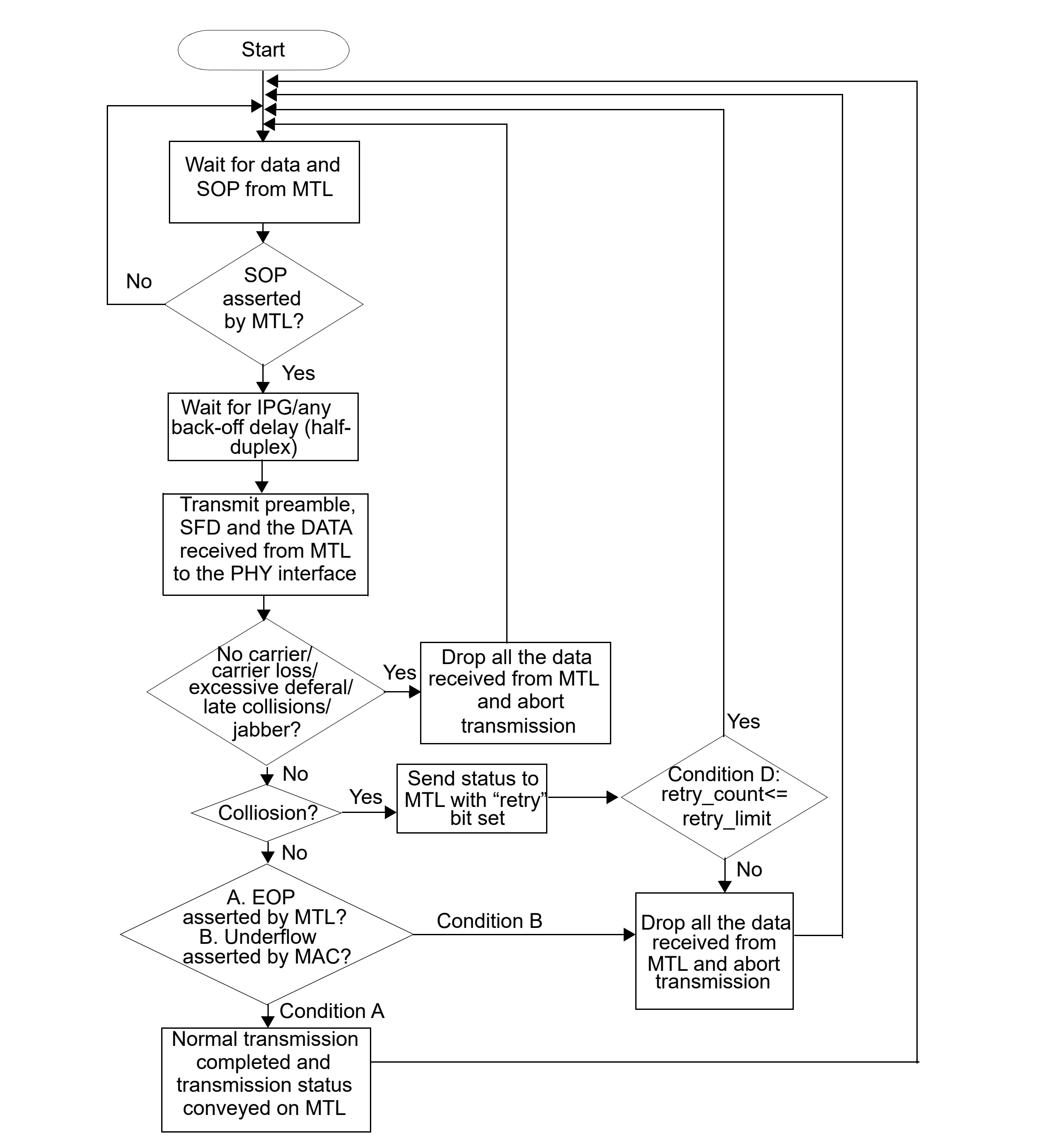
the next SOP is received. The MTL block should retransmit the same packet from SOP on observing a Retry request (in the Status) from the MAC

5. The MAC aborts the packet transmission under the following conditions:
   a) No carrier (half-duplex mode)
   b) Loss of carrier (half-duplex mode)
   c) Excessive deferral (half-duplex mode)
   d) Late collisions (half-duplex mode)
   e) Jabber

   The MAC accepts and drops all further data until the next SOP is received
6. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received
7. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet

Figure 820 illustrates the MAC transmission process flow.

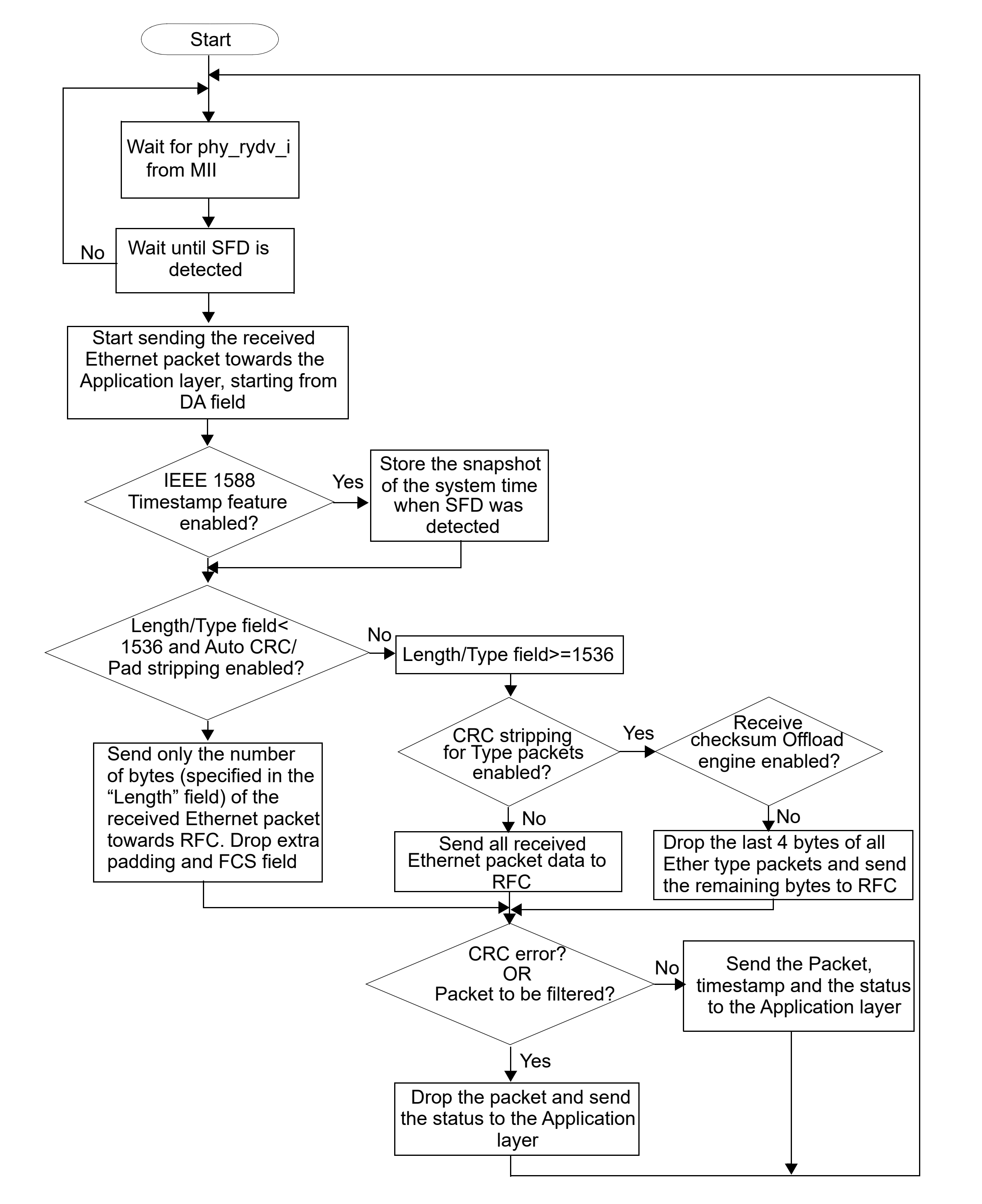
Figure 820. Overview of MAC Transmission Process Flow



47.3.4.2 MAC Reception

A receive operation is initiated when the MAC detects an SFD on MII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

Figure 821. Overview of MAC RX Operation



47.3.5 Multiple Channels and Queues Support

The Ethernet module supports 2- queues and 2-channels on Tx and Rx paths. *Figure 593* shows the architecture of the Ethernet module with multiple queues and channels. This section describes how the Ethernet module supports multiple queues and channels. It contains the following sub-sections:

- DMA with Multiple Channels
- MTL with Multiple Queues

47.3.5.1 DMA with Multiple Channels

This section describes how multiple queues are supported in the Transmit and Receive DMA paths.

47.3.5.1.1 Multi-Channel DMA Transmit Path

The Ethernet module supports 2- Tx channels with the following priority schemes:

- Fixed Priority Scheme
- Weighted Strict Priority
- Weighted Round Robin

47.3.5.1.1.1 Fixed Priority Scheme

The fixed priority scheme is the default priority scheme for the DMA channels. In fixed priority scheme, the channel with highest priority always wins the arbitration when it requests the bus. Fixed Priority Scheme for DMA Channels provides information about the priority levels of DMA channels.

Table 879. Fixed Priority Scheme for DMA Channels

Priority Level	Channel
0 (low)	Channel 0
1 (high)	Channel 1

47.3.5.1.1.2 Weighted Strict Priority

In Weighted Strict Priority (WSP), the weight corresponds to the number of burst transfers given to a channel in one arbitration cycle. The unused burst transfers of one or more channels are reallocated based on the priority. The channel with highest priority gets the unused burst transfer time before it is allocated to a channel with next highest priority.

The channel priorities are fixed. Channel 1 has the highest priority and Channel 0 has the lowest priority. When a channel uses the allocated burst transfers, the channel with next lower priority is processed. After processing the allocated bandwidth of all channels that had packets to transmit, any unused burst transfer time is allocated to the channel of the highest priority (if required), and then next highest priority (if required), and so on.

47.3.5.1.1.3 Weighted Round Robin

In Weighted Round Robin (WRR), all channels are serviced in round-robin order according to the weights settings. The TCW field of the DMA_CHIn Weighted Round Robin (WRR), all channels are _Tx_Control register provides the weight for each Transmit channel as shown in Weights for DMA Channels.

Table 880. Weights for DMA Channels

TCW Field	Transmit Channel Weight
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

The configured weights correspond to the number of burst transfers given to a channel in one arbitration cycle. The unused or excess burst transfers are distributed equally to all channels.

47.3.5.1.2 Multi-Channel DMA Receive Path

The Ethernet module supports 2- Rx channels. In the Rx direction, the MTL Rx Controller selects the Rx DMA for which it is transferring or reading the data from the Rx FIFO memory. This scheduling is based on the programming done in the respective MTL_RxQ[x]_Control register.

Each Rx DMA indicates when it is ready to transfer data and the size of the burst-length (number of beats) that it has to transfer. The scheduler checks whether sufficient data (of requested burst length) is available to be transferred to these DMAs and then selects the Rx DMA that gets serviced using the programed priorities.

47.3.5.1.3 Priority Scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channels for accessing descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the register specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a channel.

If you have enabled the Tx DMA and Rx DMA of a channel, you can specify which DMA gets the bus when the channel gets the control of the bus. You can set the priority between the corresponding Tx DMA and Rx DMA by using the TXPR field of the register. For round-robin arbitration, you can use the PR field of the register to specify the weighted priority between the Tx DMA and Rx DMA. Table 5-1 Priority Scheme for Tx DMA and Rx DMA5-1 provides information about the priority scheme between Tx DMA and Rx DMA.

Table 881. Priority Scheme for Tx DMA and Rx DMA(1)

Bit 17	Bit 18	Bit 19	Bit 20	Bit 30	Priority Scheme
x	x	x	0	1	Rx always has priority over Tx
0	0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests
0	0	1	0	0	Rx has priority over Tx in ratio 2:1
0	1	0	0	0	Rx has priority over Tx in ratio 3:1
0	1	1	0	0	Rx has priority over Tx in ratio 4:1
1	0	0	0	0	Rx has priority over Tx in ratio 5:1
1	0	1	0	0	Rx has priority over Tx in ratio 6:1
1	1	0	0	0	Rx has priority over Tx in ratio 7:1
1	1	1	0	0	Rx has priority over Tx in ratio 8:1
x	x	x	1	1	Tx always has priority over Rx
0	0	0	1	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests
0	0	1	1	0	Tx has priority over Rx in ratio 2:1
0	1	0	1	0	Tx has priority over Rx in ratio 3:1
0	1	1	1	0	Tx has priority over Rx in ratio 4:1
1	0	0	1	0	Tx has priority over Rx in ratio 5:1
1	0	1	1	0	Tx has priority over Rx in ratio 6:1
1	1	0	1	0	Tx has priority over Rx in ratio 7:1
1	1	1	1	0	Tx has priority over Rx in ratio 8:1

1. Bits 17, 18, 19, 20 and 30 are not valid if one of the paths (Transmit or Receive) is not present in a channel. The Transmit and Receive paths are always present in Channel 0.

47.3.5.1.4 Slot Number Function

When the AV feature is enabled, you can use the slot number function to schedule the data fetching from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals. You can program the slot number at which the DMA should fetch the data from system memory in the Transmit Descriptor Word 3 (TDES3). This 4-bit field allows the application to schedule data up to 16 slots of 125 µs each. This field is applicable only for the AV channels.

When the DMA fetches a Tx descriptor, it compares the slot number of the Tx descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated every 125 µs of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the second’s field of the system time is incremented, that is, the sub-second counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state till there is a match.

To enable the DMA to fetch the data only if it matches the current slot or the next two slots, you can program Bit 31 of the Slot Function Control and Status register of the corresponding DMA channel.

You can enable the check for slot number by setting Bit 31 of the Slot Function Control and Status register of corresponding DMA channel. When this check is not enabled, the packets are fetched immediately after the descriptor is read. In addition, Bits [12:15] indicate the value of the reference slot number in DMA.

Note: If the slot number in the descriptor is less than the reference slot number, the DMA takes it as a future slot.

47.3.5.1.5 Rx Queue to DMA RX Channel Mapping

The packets in the MTL Rx Queues can be routed to any one of the 2-RX DMA channels by programming the MTL_RxQ_DMA_Map0 Register. The following types of Rx Queue to DMA mapping is possible through programming:

- Static Mapping
- Dynamic (Per Packet) Mapping

47.3.5.1.5.1 Static Mapping

In this mode, all the packets of an Rx Queue is connected to a specific DMA channel. For example, all the packets from Rx Queue 0 can be routed to a DMA channel by programming Q0MDMACH (bit[28:31]) and Q0DDMACH (bit[24] = 0) of the MTL_RxQ_DMA_Map0 Register.

Similarly, packets from other Rx Queues can be routed to any DMA channel by programming register fields corresponding to each Queue.

47.3.5.1.5.2 Dynamic (Per Packet) Mapping

In this mode, the destination DMA channel of a packet being read from an Rx Queue is not constant but decided independently for each packet. For example, if you set the Q1DDMACH bit of the MTL_RxQ_DMA_Map0 register, the static mapping is disabled for Rx Queue 1 and the value in Q1MDMACH is ignored. The destination DMA channel is decided by the MAC Core receiver for each packet, depending on the following in decreasing order of priority:

1. L3-L4 filter Based DMA Selection: When the Layer 3 and Layer 4 Packet Filter is enabled, the TCP/UDP and IP header fields of the received packet are matched against the corresponding values programed and enabled for comparison in the MAC_L3_L4_Address_Control register. If the match is successful, the DMA channel number programed in the DMCHN field of the MAC_L3_L4_Address_Control register is selected as the destination DMA channel number provided DMCHEN bit of the same register is set. If the L3-L4 Registers do not give a comparison match, then the Ethernet module proceeds to the next step below
2. Ethernet DA-Based DMA Selection: The DA address of the received packet is compared against the programed DA values in MAC Address Registers. If the address matches any of the programed values, the corresponding DCS field (when enabled) determines the destination DMA channel number

If none of the above operations is able to make a successful match/decision, then the packet is routed to DMA Channel 0 by default

47.3.5.1.6 Selection of Tag Priorities assigned to Tx and Rx Queues

The VLAN Tag priorities can be assigned to Tx Queues by programming the corresponding PSTQIn field in the *Section 47.2.14: Transmit Queue Priority Mapping 0 Register (MAC_TXQ_PRTY_MAP0)* (for Tx Queues 0 and 1).

The bit corresponding to the VLAN Tag priority can be set in the PSTQIn field for assigning that priority to the Tx Queue. For example, if you want to assign VLAN Tag priority of 3 to Tx Queue 0, set bit [3] in PSTQ0 field.

The same VLAN Tag priority can be set for multiple Tx queues. The Tx packet association to particular Tx Queue is governed by the application, so the MAC does not route the packet based on Tx Queue priority mapping; it is only used for Pause Flow Control (PFC). The settings in the PSTQ*n* field determines the Tx Queues blocked for transmission when the PFC packet is received with corresponding VLAN Tag priorities enabled.

The VLAN Tag priorities can be assigned to Rx Queues by programming the PSRQ field in the corresponding MAC_RxQ_Ctrl2 register. The bit corresponding to the VLAN Tag priority can be set in the PSRQ field for assigning that priority to the Rx Queue. For example, if you want to assign VLAN Tag priority of 3 to Rx Queue 0, set bit [3] in PSRQ field of MAC_RxQ_Ctrl2 register. The VLAN Tag priority assigned to particular Rx Queue must be unique, that is, more than one Rx Queue cannot be assigned the same VLAN Tag priority. However, more than one VLAN Tag priorities can be assigned to same Rx Queue.

The settings in the PSRQ field are used for VLAN Tagged Rx packet routing to Rx Queues as well as for PFC based Tx flow control. The received VLAN Tagged Rx packet is routed to Rx Queue that has the VLAN Tag priority match. In PFC based Tx flow control, PSRQ field corresponding to a particular Rx Queue is used for enabling VLAN Tag priorities in the PFC packet transmitted when corresponding Rx Queue threshold levels are reached.

47.3.5.1.7 Rx Side Routing from MAC to Queues

The MAC routes the Rx packets to the Rx Queues based on the following packet types:

- VLAN Tag Priority field in VLAN Tagged AV data packets
- AV Control packets
- VLAN Tag Priority field in VLAN Tagged packets
- Untagged IEEE1588 PTP over Ethernet packets

The outer C-VLAN Tagged AV data Rx packets can be routed based on the priorities assigned to Rx Queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 register and the corresponding Rx Queue is enabled for AV through RXQ*n*EN field in MAC_RxQ_Ctrl0 register. These packets may be single VLAN Tagged with C-VLAN type or Double VLAN Tagged with outer VLAN Tag of C-VLAN type when Double VLAN feature is enabled (EDVLP bit in MAC_VLAN_Tag register is set) with inner C-VLAN Tagged or inner S-VLAN Tagged when SVLAN processing is enabled (ESVL bit in MAC_VLAN_Tag register is set). This type of Rx packet routing is available when AV feature and Multiple Rx Queues are selected in the configuration.

The AV Control Rx packets can be routed based on the Rx Queue number specified in the AVCPQ field in MAC_RxQ_Ctrl1 register and corresponding Rx Queue is enabled for AV through RXQ*n*EN field in MAC_RxQ_Ctrl0 register. These packets may be single VLAN Tagged with C-VLAN type or Double VLAN Tagged with outer VLAN Tag of C-VLAN type when Double VLAN feature is enabled (EDVLP bit in MAC_VLAN_Tag register is set) with inner C-VLAN Tagged or inner S-VLAN Tagged when SVLAN processing is enabled (ESVL bit in MAC_VLAN_Tag register is set). This type of Rx packet routing is available when AV feature and Multiple Rx Queues are selected in the configuration.

The VLAN Tagged Rx packets can be routed based on the priorities assigned to Rx Queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 register and the corresponding Rx Queue is enabled for generic through RXQ*n*EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when Multiple Rx Queues are selected in the configuration.

The untagged IEEE1588 PTP over Ethernet Rx packets can be routed based on the Rx Queue number specified in the AVPTPQ field in MAC_RxQ_Ctrl1 register and the corresponding Rx Queue is enabled for AV through RXQ*n*EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when AV feature and Multiple Rx Queues are selected in the configuration.

If the Rx packet cannot be classified in any of the defined packet type for routing, it will be routed through Rx Queue 0.

47.3.5.1.8 Rx Side Arbitration between DMA and MTL

After completing the current packet processing, the DMA Channel Controller fetches the next descriptor. After the descriptor fetching is complete, the DMA Channel Controller evaluates the amount of data to be to be transferred to the Rx Buffer based on the programed PBL and Rx Buffer Length. Accordingly, it requests the MTL to transfer the data.

After servicing the current request, the MTL Rx Queue arbitration scheme selects Rx Queue based on the arbitration scheme (RAA field of the MTL_Operation_Mode register) and the weights programed in the Queue <n> Receive Control Register. The arbitration is done among Queues for which DMA is ready to service. After the Rx Queue is selected, PBL (Programmable Burst Length) amount of data is read out from that Queue and is routed to the Rx DMA Channel based on the Rx Channel selection criteria.

The arbitration takes place when every PBL data transfer is completed and descriptors are ready for the processing from at least one DMA Channel.

47.3.5.2 MTL with Multiple Queues

The Ethernet module supports 2xTx and 2xRx queues. The number of Tx queues is equal to the number of DMA channels. You can control a Tx queue through *Section 47.2.169: Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE)*. You can control an Rx queue through *Section 47.2.175: Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE)*.

47.3.5.2.1 Queue Memory

All MTL Tx queues share one Tx memory and all Rx queues share one Rx memory with programmable individual queue sizes in blocks of 256 bytes. The application can program the Tx queue size in the TQS field of *Section 47.2.169: Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE)* and the Rx queue size in the RQS field of the *Section 47.2.175: Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE)*.

47.3.5.2.2 Queue Modes

You can enable a Tx queue for generic, AV, or all types of traffic. Generic Tx queues use WRR or WSP queuing algorithms. AV Tx queue uses CBS or SP queuing algorithms.

You can enable an Rx queue for generic, AV, or all types of traffic. A particular queue enabled for generic or AV based routing is determined by the RXQ*n*EN field of corresponding queue. The following list explains how Rx queues are enabled based on the selected features:

- Multiple Generic Rx queues: All queues are enabled for generic queuing based on the VLAN Tag priority. The VLAN Tag priority should match the PSRQ field of the MAC_RxQ_Ctrl2 register. Untagged packets are routed through Queue 0. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register
- Multiple Rx queues with AV feature: When you select the *Enable Audio Video Bridging* option, AV is enabled for all selected Rx queues. The queuing is done based on the VLAN Tag priority. The VLAN Tag priority should match the PSRQ field of the MAC_RxQ_Ctrl2 register. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register. The AV control packets (tagged or untagged) are routed based on the AVCPQ field of the MAC_RxQ_Ctrl1 register. The PTP over Ethernet packets are routed based on the AVPTPQ field of MAC_RxQ_Ctrl1 register

47.3.5.2.3 Queue Priorities

You can program the priority of a Tx queue in the MAC_TxQ_Prty_Map0 register. You can program the priority of an Rx queue in Bits[16:23] of corresponding Rx Flow Control Register. The priority should be assigned in the following order:

a) AV queue (high priority)

b) Best-effort queue (low priority)

The software should put packets with correct priorities in the respective programed queue on Tx side. The MAC uses the programed priorities for blocking the Tx queues when a PFC packet is received. If a single queue is selected for multiple priorities and PFC is enabled, the entire queue is paused if one or more priorities in the queue are paused.

47.3.5.2.4 AV Feature

The Ethernet module supports the AV data transfer in 100 Mbps mode. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

- IEEE 802.1Qav-2009: Allows the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources
- IEEE 802.1Qat-2009: Allows the network resources to be reserved for specific traffic streams traversing a bridged local area network
- IEEE 802.1AS-2011: Specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration

47.3.5.2.4.1 Transmit Path Functions

When you select the AV feature, the Transmit paths of Queue 0 and the highest enabled queue are enabled by default.

The Transmit path of Queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of size equal to the programed threshold limit.

The Transmit paths of additional queues support traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

- The queue contains one or more packets
- The credit for the queue is positive as per the algorithm

You can disable the credit-based shaper algorithm for all queues or lower-priority queues. When you disable the credit-based shaper algorithm for a queue, the channel uses the default strict priority algorithm.

Each Transmit DMA has a separate descriptor chain for fetching the transmit data. The Transmit channel that gets the access to the system bus depends on the DMA arbiter.

The Transmit path has a shared FIFO (MTL layer) for each queue, as shown in *Figure 593*. The data fetched by the DMA is put in the respective part of the FIFO. The MTL Tx Queue Scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for a queue, the corresponding queue is selected for transmission if the following conditions are true:

- If the packet is available in the channel and has a positive or zero credit
- If the higher priority queue has no packet waiting in the FIFO

If the credit-based shaper algorithm is disabled for all queues, the packet to be transmitted from a queue is selected based on the priority scheme described in Fixed Priority Scheme for DMA Channels.

47.3.5.2.4.2 Receive Path Functions

When you select the AV feature, the Receive path of Queue 0 is enabled by default. All traffic is received on this channel. You can enable the Receive paths of additional queues. By enabling the Receive paths of multiple channels, you can demultiplex the received data to send the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if it is an AV packet and its corresponding VLAN Priority tag value. This status is updated in the Extended Status field of the Receive descriptor as explained in *Section 47.4.1.4: Receive Descriptor*. All received packets with the EtherType field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

- AV data packets: The AV data packets are always tagged. The tagged AV data packets are received based on the programed priority value. To specify the channel to which an AV packet with a given priority must be sent, program Bits[16:23] in the Transmit Flow Control Register of the corresponding queue
- AV control packets: The AV control packets can be either tagged or untagged. The untagged AV control packets are received on Queue 0 by default. To receive these

packets on any other queue, you can program Bits[29:31] of the MAC_RxQ_Ctrl1 register. Similar to the AV data packets, the tagged AV control packets are received based on the programed priority value

In addition to the AV packets, you can receive the untagged PTP packets on any queue. By default, the PTP packets (tagged or untagged) are received on Queue 0. To receive these packets on any other queue, you need to program Bits[25:27] of the MAC_RxQ_Ctrl1 register.

47.3.5.2.4.3 Credit-Based Shaper Algorithm

The MTL Queue Scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in Queue 0. You can program the additional queues to use the credit-based shaper algorithm.

The following sections provide information about how you can implement the credit-based shaper algorithm:

Credit Value

The credit value is accumulated every transmit clock cycle, that is, 40 ns for 100 Mbps and 8 ns for 1000 Mbps. The credit to be added or subtracted per cycle can be fractional based on the required idleSlope and sendSlope values, as described in Credit Value Per Transmit Cycle Example.

Table 882. Credit Value Per Transmit Cycle Example

Mode	Values	Description
100 Mbps	– portTransmitRate = 100 Mbps – idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) – sendSlope = 30 Mbps	– credit = 2.8 bits accumulates per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted. – credit = 1.2 bits drains per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO based on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the AMBA AHB application bus.

The credit for a queue builds up only when the packet is available but it cannot be transmitted because the MAC is sending a packet from another queue. The Ethernet module supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. You can enable this mode with Bit 30 of the CBS Control registers of corresponding queues.

When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet waiting in a queue and other queues are transmitting. When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit Values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programed values should be the credit accumulated or drained per clock cycle scaled by 1024, such as, 2.8 x 1024 = 2867 and 1.2 x 1024 = 1229. In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the values to be programed in the hiCredit and loCredit registers of the corresponding channel are 12000 x 1024 bits and two's complement of 3036 x 1024, respectively.

Bandwidth Status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register gives the average number of bits transmitted during the previous programed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programed in Bits[25:27] of the CBS control register of the respective queue. For example, if you have programed two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.

A lower priority queue, which is using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue, which is using the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, Queue 1 and Queue 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25%, respectively) and Queue 0 uses the strict-priority algorithm. If Queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by Queue 0. Queue 2 cannot exceed the reserved bandwidth of 25%.

47.3.5.2.5 Tx Side Arbitration between DMA and MTL

Because the number of Tx DMA channels is always equal to the number of Tx Queues in the MTL, they are always mapped directly. For instance, each Tx DMA pushes data into its respective Tx Queue assigned to it. The main reason for this strategy is to optimize and increase the efficiency of the DMA data transfer, as well as to simplify the TX Checksum Engines in the MTL.

The data inside each Tx Queue is stored in packets. Hence, if two DMAs are allowed to transfer data into the same queue, when a Tx DMA starts a packet transfer, the other DMA cannot transfer data unless the previous packet is completely pushed-in. This means that the second DMA remains idle until the first packet is transferred. Hence, each DMA is always connected directly to its corresponding Tx Queue.

47.3.6 Double VLAN Processing

The Ethernet module supports the double VLAN tagging feature in which the MAC can process up to two VLAN tags (inner and outer). The Ethernet module supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status

47.3.6.1 VLAN Processing in Transmit Path

Table 883 describes the double VLAN processing features supported by the MAC on the Transmit side.

Table 883. Double VLAN Processing Features in Tx Path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of MAC_VLAN_Incl and MAC_Inner_VLAN_Incl registers. The Ethernet module supports processing of any sequence of outer and inner VLAN tags. **Note:** The Ethernet module does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.
VLAN Tag deletion	You can enable the VLAN tag deletion for outer or inner tag through VLC field in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	You can enable the VLAN tag insertion or replacement for outer or inner tag through VLC field in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register is used to determine whether the VLAN tag should be taken from the register or the Control Word.

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields. Insertion or replacement is done based on the setting of VLTI bit in the MAC_VLAN_Incl register as described in *Table 884*

Table 884. VLAN Insertion or Replacement Based on VLTI Bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: – VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of MAC_VLAN_Incl register) – VLAN Tag field with the Content of the VT field of Transmit context descriptor of the packet.
VLTI bit is reset	The MAC inserts or replaces the following: – VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of MAC_VLAN_Incl register) – VLAN Tag field with the VLT field of MAC_VLAN_Incl register

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88a8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- Enabling VLAN insertion, replacement, or deletion for all packets. To enable this feature for all packets, program the VLC and VLP fields of MAC_VLAN_Incl register
- Enabling VLAN insertion, replacement, or deletion for selective packets. To enable this feature for selective packets, Program the VTIR field of TDES2 Normal Descriptor

47.3.6.2 VLAN Processing in Receive Path

Table 885 describes the features supported by the MAC on the Receive side and the corresponding bits in the MAC_VLAN_Tag register.

Table 885. Double VLAN Processing in Rx Path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.

Table 885. Double VLAN Processing in Rx Path (continued)

Feature	Description
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

47.3.7 Source Address Insertion, Replacement, or Deletion

Software can use the SA insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address Registers in the SA field
- Replace the content of the SA field with the content of the MAC Address Registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

You can enable the SA insertion or replacement feature for all Transmit packets or selective packets:

- To enable this feature for all packets, program the SARC field of the MAC_Configuration register
- To enable this feature for selective packets, use the following method:
  Program the SA Insertion Control field (Bits[6:8] of TDES3 Normal Descriptor (Read Format)) in the first Transmit descriptor of the packet. When Bit 6 of TDES3 is set, the SA Insertion Control field indicates insertion or replacement by MAC Address1 registers. When Bit 6 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers
- If both MAC_Configuration Register and TDES3 are programed for SA insertion, then setting of MAC_Configuration register takes precedence

If MAC Address1 Registers are not enabled, the MAC Address0 registers are used for insertion or replacement irrespective of the value of the most-significant bit of the SA Insertion Control field.

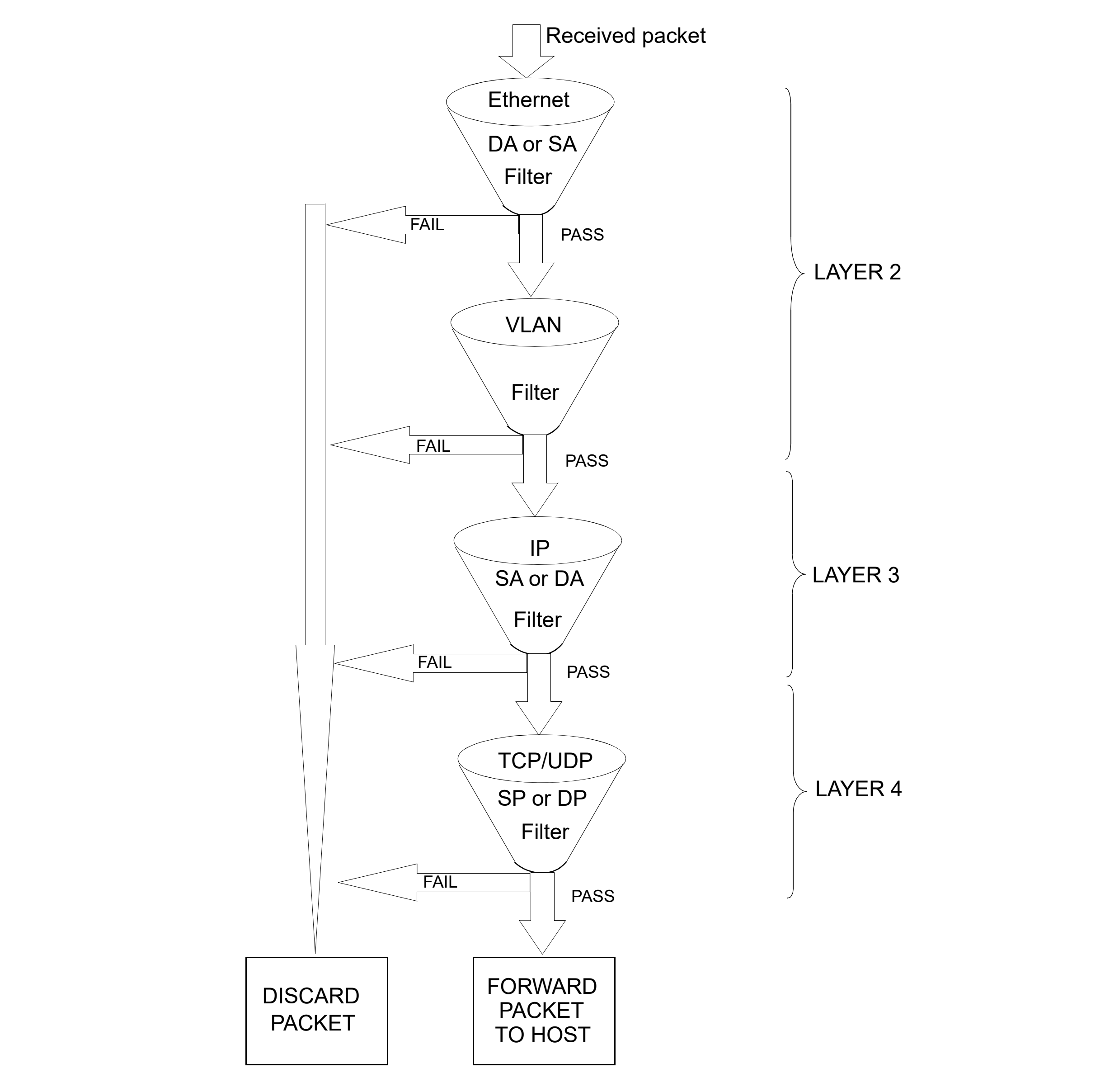
47.3.8 Packet Filtering

The Ethernet module supports the following types of filtering for Rx packets:

- Source Address or Destination Address Filtering: The Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet
- VLAN Filtering: The Ethernet module supports the VLAN tag-based and VLAN hash filtering
- Layer 3 and Layer 4 Filtering: Layer 3 filtering refers to source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering

Figure 822 shows the filtering sequence for Rx packets.

Figure 822. Packet Filtering Sequence



47.3.8.1 Source Address or Destination Address Filtering

The Address Filtering Module of the MAC checks the source address (SA) and destination address (DA) fields of each incoming packet.

47.3.8.1.1 Unicast Destination Address Filtering

The MAC supports up to 128 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of MAC_Packet_Filter register is reset), the MAC compares all 48 bits of received unicast address with the programed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr127 addresses are selected with an individual enable bit. For MacAddr1 to MacAddr31 addresses, you can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in the register. This enables group address filtering for the DA. The MacAddr32 to MacAddr127 addresses do not have mask control and all 6-bytes of the MAC address are compared with the received 6-bytes of DA.

In hash filtering mode (when HUC of MAC_Packet_Filter register bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects Bit 63 of selected register, and a value of 11111 selects Bit 0 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

47.3.8.1.2 Multicast Destination Address Filtering

To program the MAC to pass all multicast packets, set the PM bit in *Section 47.2.3: MAC Packet Filter Register (MAC_PACKET_FILTER)*. If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the MAC_Packet_Filter register.

In Perfect filtering mode, the multicast address is compared with the programed MAC Destination Address registers (1–31). Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects Bit 63 of selected register and a value of 111111 selects Bit 0 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

47.3.8.1.3 Hash or Perfect Address Filtering

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in *Section 47.2.3: MAC Packet Filter Register (MAC_PACKET_FILTER)*. This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

47.3.8.1.4 Broadcast Address Filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in MAC_Packet_Filter register.

47.3.8.1.5 Unicast Source Address Filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programed in the SA registers. You can configure the MAC Address registers[1–31] to use SA instead of DA for comparison by setting Bit 1 of corresponding register.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in MAC_Packet_Filter register. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word. When the SAF bit is set, the SA filter and

DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

47.3.8.1.6 Inverse Filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of MAC_Packet_Filter register. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 886 and *Table 887* summarize the DA and SA filtering based on the type of packets received.

Table 886. Destination Address Filtering

Packet Type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA Filter Operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multicast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 887. Source Address Filtering

Packet Type	PR	SAIF	SAF	SA Filter Operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

Note: When the RA bit of MAC_Packet_Filter register is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx Status.

47.3.8.2 VLAN Filtering

The Ethernet module supports the following types of VLAN filtering:

- VLAN Tag Perfect Filtering
- VLAN Tag Hash Filtering

47.3.8.2.1 VLAN Tag Perfect Filtering

In VLAN tag perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of MAC_VLAN_Tag register. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of MAC_VLAN_Tag register. The VLAN packet status bit (Bit 21 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

47.3.8.2.2 VLAN Tag Hash Filtering

The MAC provides VLAN tag hash filtering with a 16-bit Hash table. The MAC performs the VLAN hash matching based on the VTHM of the MAC_VLAN_Tag register. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the perfect or hash filter, the packet should be dropped. If the VLAN perfect and VLAN hash match are enabled, a packet is considered as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and hash filters indicate mismatch.

Table 888 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of the MAC_Packet_Filter register is set, all packets are received and the VLAN match status is indicated in the VF bit of RDES2 Normal Descriptor (Write-Back Format). When the RA bit is not set and the VTFE bit is set in the MAC_Packet_Filter register, the packet is dropped if the final VLAN match status is Fail. In Table 888, value *X* means that this column can have any value.

When VLAN VID is programed to 0 in the VL field of MAC_VLAN_Tag register, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN hash match depends on the VTHM and VTIM bits in MAC_VLAN_Tag register.

Table 888. VLAN Match Status(1)

VID	VLAN Perfect Filter Match Result	VTHM Bit	VLAN Hash Filter Match Result	VTIM Bit	Final VLAN Match Status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

1. In this table, X represents any value.

Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in MAC_VLAN_Tag register. When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of MAC_VLAN_Hash_Table register. When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of MAC_VLAN_Hash_Table register.

47.3.8.3 Layer 3 and Layer 4 Filtering

The Ethernet module supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the Receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the MAC_Configuration register to enable the Rx Checksum Offload Engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched Packets: The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of MAC_Configuration register is set and one of the following conditions is true:
  - All enabled Layer 3 and Layer 4 fields match
  - At least one of the enabled field matches and other fields are bypassed or disabled
- Unmatched Packets: The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets. The aborted or partial packets can be dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programed to a small value, such that packet transfer to application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.
- Non-TCP or UDP IP Packets: By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Note: The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

47.3.8.3.1 Layer 3 and Layer 4 Filters Register Set

The MAC implements a set of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as MAC_L3_L4_Control0, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- MAC_Layer4_Address0
- MAC_Layer3_Address0_Reg0
- MAC_Layer3_Address1_Reg0
- MAC_Layer3_Address2_Reg0
- MAC_Layer3_Address3_Reg0

47.3.8.3.2 Layer 3 Filtering

The Ethernet module supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, you can enable the second and third data registers to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

47.3.8.3.3 Layer 4 Filtering

The Ethernet module supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

47.3.9 IEEE 1588 Timestamps

The IEEE 1588 defines a Precision Time Protocol (PTP) which enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

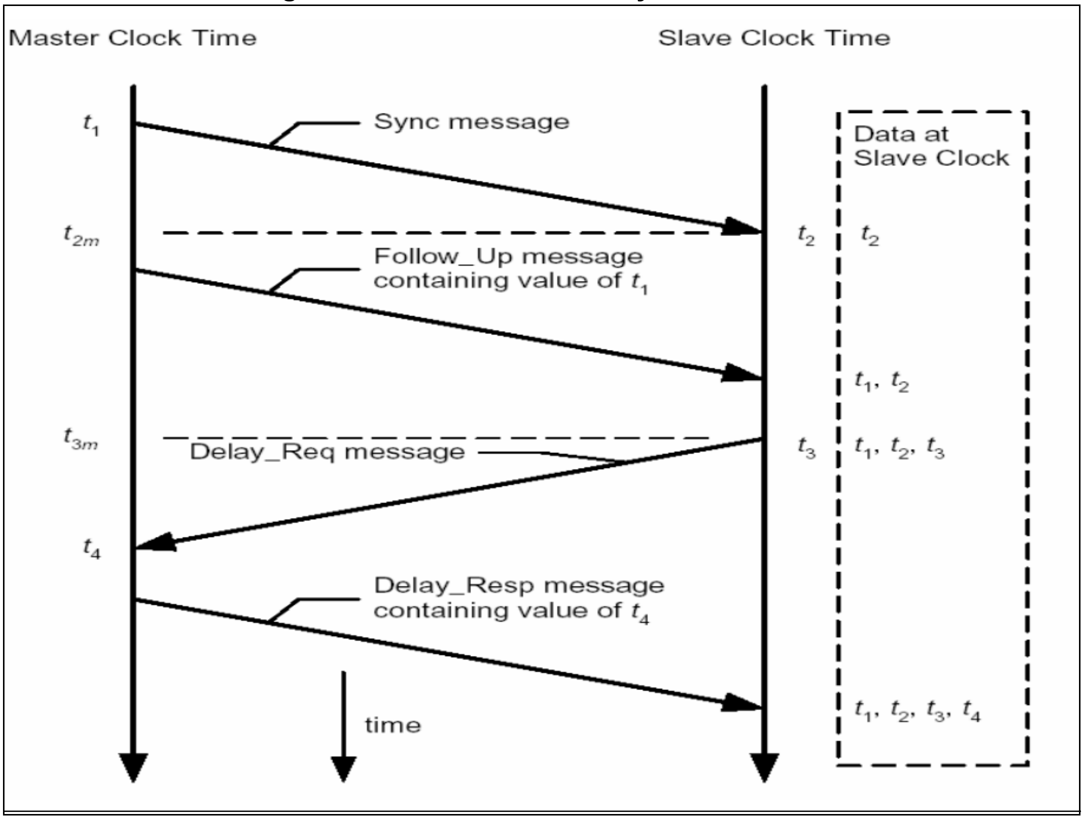
The Ethernet module supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588-2008 supports PTP transported over Ethernet. The Ethernet module provides programmable support for both standards. It supports the following features:

- Supports both timestamp formats
- Provides an option to take snapshot of all packets or only PTP type packets
- Provides an option to take snapshot of only event messages
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Provides an option to select the node to be a master or slave for ordinary and boundary clock
- Identifies the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Provides an option to measure sub-second time in digital or binary format

47.3.9.1 Delay Request-Response Mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. *Figure 823*, shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

Figure 823. Networked Time Synchronization



As shown, the PTP uses the following process:

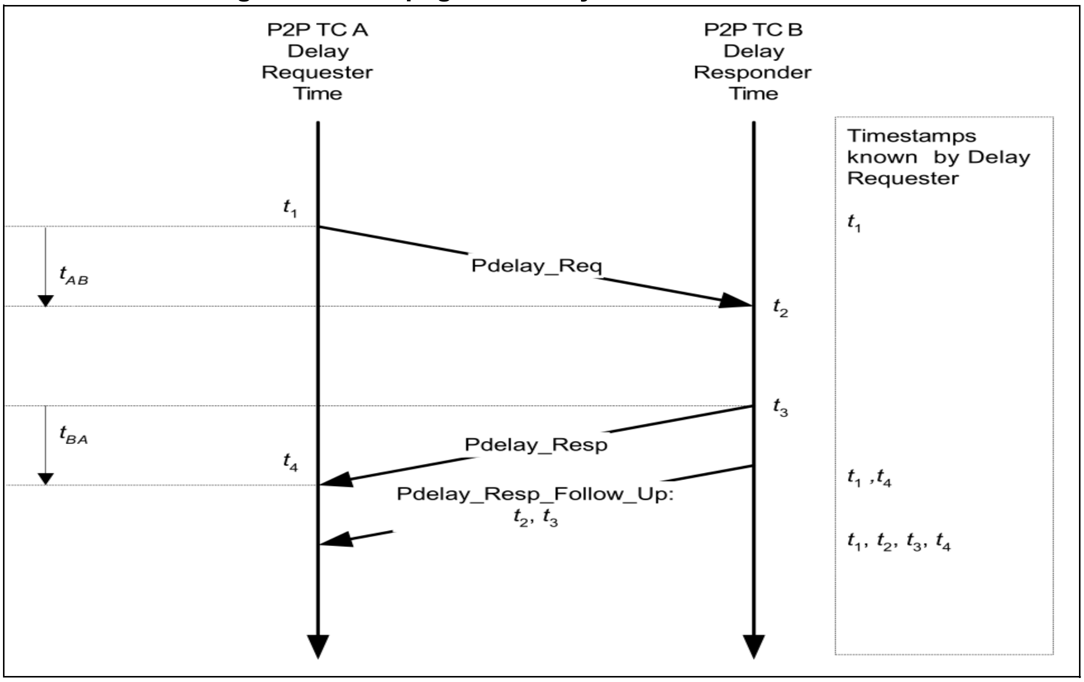
1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at t1. This time must be captured for Ethernet ports at MII
2. The slave receives the Sync message and also captures the exact time, t2, using its timing reference
3. The master sends a Follow_up message to the slave, which contains t1 information for later use
4. The slave sends a Delay_Req message to the master and notes the exact time, t3, at which this packet leaves the MII interface
5. The master receives the message, capturing the exact time t4, at which the message enters its system
6. The master sends the t4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the timing reference of the master

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

47.3.9.2 Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 supports peer-to-peer PTP (Pdelay) message in addition to the SYNC, Delay Request, Follow-up, and Delay Response messages. Figure 824 shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 824. Propagation Delay Calculation in Clocks



As shown, the propagation delay is calculated in the following way:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t1) for the Pdelay_Req message
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t2) for this message
3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t3) for this message

To minimize errors because of any frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:

   a) Difference between the timestamps t2 and t3 in the Pdelay_Resp message
   b) Difference between the timestamps t2 and t3 in the Pdelay_Resp_Follow_Up message
   c) Timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively

4. Port 1 generates a timestamp (t4) on receiving the Pdelay_Resp message
5. Port 1 uses all four timestamps to compute the mean link delay

47.3.9.3 Clock Types

The Ethernet module supports the following clock types defined in the IEEE 1588-2008:

- Ordinary Clock
- Boundary Clock
- End-to-End Transparent Clock
- Peer-to-Peer Transparent Clock

47.3.9.3.1 Ordinary Clock

The ordinary clock in a domain supports a single copy of the protocol. The ordinary clock has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

- Sends and receives PTP messages. The timestamp snapshot can be controlled
- Maintains the data sets such as timestamp values

Table 889 shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

Table 889. Ordinary Clock: PTP Messages for Snapshot

Master	Slave
Delay_Req	SYNC

For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in MAC_Timestamp_Control Register.

47.3.9.3.2 Boundary Clock

The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy, and signaling terminate in the protocol engine of the boundary clock and such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

- The clock data sets are common to all ports of the boundary clock
- The local clock is common to all ports of the boundary clock

47.3.9.3.3 End-to-End Transparent Clock

The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the Ingress port to the Egress port.

The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both Ingress and Egress ports only for the messages mentioned in the bullet points below. The snapshot is taken by setting the SNAPTYPSEL bits to ‘10’ in the MAC_Timestamp_Control register.
End to End Transparent Clock: PTP Messages for Snapshot:

- SYNC
- Delay_Req

47.3.9.3.4 Peer-to-Peer Transparent Clock

The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer. The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as mentioned in the below bullet points.
Peer-to-Peer Transparent Clock: PTP Messages for Snapshot:

- SYNC
- Pdelay_Req
- Pdelay_Resp

The snapshot is taken by setting the SNAPTYPESEL bit to ‘11’ in MAC_Timestamp_Control register.

47.3.9.4 Reference Timing Source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

The Ethernet module provides an Internal Reference Time (80-bit). It takes the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps. The timestamp has the following fields:

- UInteger48 secondsField: The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bits wide. For example, 2.000000001 seconds are represented as secondsField = 0x0000_0000_0002
- UInteger32 nanosecondsField: The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 nanoseconds are represented as nanoSeconds = 0x0000_0001

The nanoseconds field supports the following two modes:

- Digital rollover mode: In this mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds
- Binary rollover mode: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit

These modes are set through TSCTRLSSR bit in MAC_Timestamp_Control register.

The timestamp maintained in the MAC is 64-bit wide. The overflow to upper 16-bits of seconds register happens once in 130 years. You can read the values of the upper 16-bits of the seconds field from the CSR register.

The Ethernet module supports the following reference timing source features:

- Fixed Pulse-Per-Second OutputPulse-Per-Second Output
- Flexible Pulse-Per-Second OutputPulse-Per-Second Output
- Auxiliary Snapshots with External Events

47.3.9.4.1 Fixed Pulse-Per-Second Output

The Ethernet module supports the pulse-per-second (PPS) output that is given to indicate 1 second interval (default). You can change the frequency of the PPS output by setting the PPSCTRL0 field in the MAC_PPS_Control register.

47.3.9.4.2 Flexible Pulse-Per-Second Output

The Ethernet module also provides the flexibility to program the start or stop time, width, and interval of the pulse generated on the ptp_pps_o output. The Ethernet module provides the following features with the flexible PPS output:

- Supports programming the start or stop time in terms of system time
- Supports programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time
- Supports programming the stop time in advance, that is, you can program the stop time before the actual start time has elapsed
- Supports programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of sub-second increment value programed in the MAC_Sub_Second_Increment register. You can program the width of pulse from 1 to 2^{32}-1 units of sub-second increment value
- Supports programming the interval, between the rising edges of PPS signal, in terms of number of units of sub-second increment value. You can program the interval between pulses from 1 to 2^{32}-1 units of sub-second increment value
- Provides the option to cancel the programed PPS start or stop request
- Indicates error if the start or stop time being programed has already elapsed

Note: The PTP Reference clock mentioned in the following sections is the clock at which the system time gets updated. When the TSCFUPDT bit of MAC_Timestamp_Control register is set to 0, this clock is similar to the clk_ptp_ref_i clock. In the Fine Correction mode, this is the clock tick at which the system time gets updated (using incr_sub_sec_reg shown in Figure 825).

47.3.9.4.2.1 PPS Start or Stop Time

You can initially program the start time in the Target Time register. If required, you can again program the start or stop time but you can do it only after the earlier programed value is synchronized to the PTP clock domain. Bit 0 of MAC_PPS*n*_Target_Time_Nanoseconds register indicates that the synchronization is complete. This enables you to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, you should program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the

MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

47.3.9.4.2.2 PPS Width and Interval

The PPS width and interval are programed in terms of granularity of system time, that is, number of the units of sub-second increment value. For example, to have a PPS pulse width of 40 ns and interval of 100 ns with PTP reference clock of 50 MHz, you should program the width and interval to values 2 and 5, respectively. You can achieve smaller granularity by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, you should program or update the interval and width of the PPS signal output.

47.3.9.4.3 Auxiliary Snapshots with External Events

The auxiliary snapshot feature allows you to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the ptp_aux_ts_trig_i sideband signal.

The Ethernet module has two auxiliary snapshot inputs and depth 4 of a single common auxiliary snapshot FIFO. The snapshots taken for any input are stored in a common FIFO. The application can read the MAC_Timestamp_Status register to know the timestamp of which input is available for reading at the top of this FIFO.

The MAC stores these snapshots in a FIFO. Only 64-bits of the timestamp are stored in the FIFO. You can read the upper 16-bits of seconds from the MAC_System_Time_Higher_Word_Seconds register. When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the MAC_Timestamp_Status register. This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in MAC_Auxiliary_Control register. When multiple snapshots are present in the FIFO, the count is indicated in Bits[4:6] of MAC_Timestamp_Status register.

47.3.9.5 System Time Register Module

The 64-bit time is maintained in this module and updated using the input reference clock (clk_ptp_ref_i). This time is the source for taking snapshots (timestamps) of Ethernet packets being transmitted or received at the MII interface.

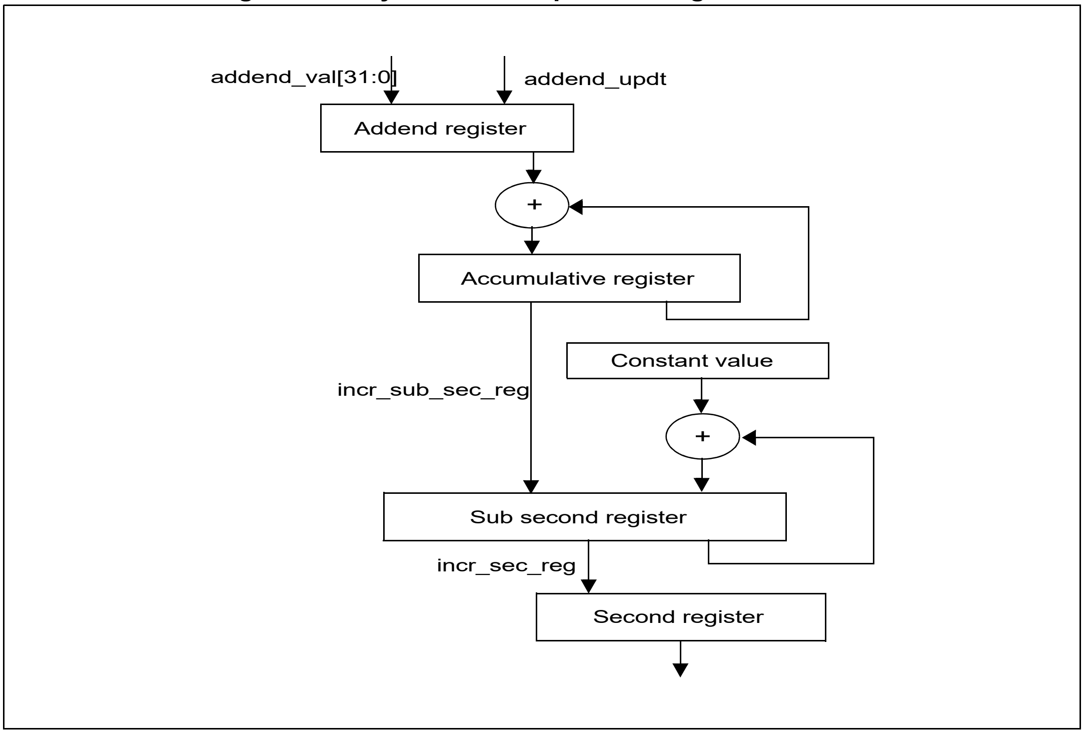
The system time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the Timestamp Update register. For initialization, the system time counter is written with the value in the Timestamp Update register. For system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, the frequency drift of a slave clock (clk_ptp_ref_i) with respect to the master clock (as defined in IEEE 1588-2002) is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register, as shown in *Figure 825*. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

Note: You must connect a PTP clock with a frequency higher than the frequency required for the specified accuracy.

This algorithm is shown in *Figure 825*.

Figure 825. System Time Update Using Fine Method



The System Time Update logic requires a 50-MHz clock frequency to achieve 20-ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz / 50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, or 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC4EC4. If the clock drifts higher, for example, to 67 MHz, the addend register must be set to 0xBF0B7672. When the clock drift is nil, the default addend value of 0xC1F07C1F (2^{32} / 1.32) must be programed.

In 5-12, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20 ns steps). Two different methods are used to update the System Time register depending on the configuration.

The software must calculate the drift in frequency based on the Sync messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$FreqCompensationValue_0 = 2^{32} / FreqDivisionRatio$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

- At time $MasterSyncTime_n$ the master sends the slave clock a Sync message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as
  $MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$
- The master clock count for current Sync cycle, $MasterClockCount_n$ is
  $MasterClockCount_n = MasterClockTime_n – MasterClockTime_{n-1}$ (assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)
- The slave clock count for current Sync cycle, $SlaveClockCount_n$ is
  $SlaveClockCount_n = SlaveClockTime_n – SlaveClockTime_{n-1}$
- The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is
  $ClockDiffCount_n = MasterClockTime_n – SlaveClockTime_n$
- The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is
  $FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$
- The frequency compensation value for Addend register, $FreqCompensationValue_n$ is
  $FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n-1}$

In theory, this algorithm achieves lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value from the master, the algorithm corrects it at the cost of more Sync cycles.

47.3.9.6 Transmit Path Functions

The MAC captures a timestamp when the Start Packet Delimiter (SFD) of a packet is sent on the MII interface. The packets, for which you want to capture timestamps, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. You need to specify the packets for which you want to capture timestamps. You can do this by using the control bits in the Transmit descriptor. The MAC returns the timestamp to the software inside the corresponding Transmit descriptor, thus connecting the timestamp automatically to the specific PTP packet. The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

47.3.9.7 Receive Path Functions

The MAC can be programed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. You can control the snapshot of the time to be sent to the application by using the following options of the MAC_Timestamp_Control register.

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type

This feature controls the type of messages for which snapshots are taken.

The DMA returns the timestamp to the software inside the corresponding Receive Descriptor. The extended status, containing the timestamp message status and the IPC status, is written in normal descriptor RDES1 and the snapshot of the timestamp is written in RDES0 and RDES1 fields of context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Note: The Ethernet module also supports the PTP messages over VLAN packets.

47.3.9.8 Timestamp Correction

According to the IEEE 1588 specification, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network. Because the reference timing source (the PTP clock ***clk_ptp_ref_i***) is different from the MAC Tx or Rx clock, the captured timestamp must be corrected for latency issues because of synchronization. In addition, latency issues between the internal snapshot point and the recommended capture point (the boundary between the node and the network), must also be corrected.

47.3.9.8.1 Ingress Correction

In the Receive side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (Ingress Correction Value) programed in the Ingress Correction register. The value that needs to be programed in the ingress correction register is calculated as mentioned below:

The timestamp correction because of synchronization is compensated by adding INGRESS_SYNC_CORR to the synchronized timestamp value.

```
INGRESS_SYNC_CORR = -(2*PTP_CLK_PER)
```

The latency correction between the message timestamp point and the internal timestamp snapshot point is done by subtracting the latency value (INGRESS_LATENCY) with the captured timestamp.

```
Ingress Correction = INGRESS_SYNC_CORR - INGRESS_LATENCY
```

Ingress correction is performed by programming the TSIC field in the MAC Timestamp Ingress correction register. The ingress correction is always negative and has a unit of nanoseconds. The value is represented in complement form.

When TSCTRLSSR bit in MAC_Timestamp_Control register is set, this has an accuracy of 1 ns.

It is represented by setting bit 0 to 30 and bits 1:31 containing 10^9 - <ingress_correction_value> represented in binary. For example, if the required correction value is -5 ns, then the programed value is 0xBB9A_C9FB.

When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 ns. It is represented by setting bit 0 to 30 and bits 1:31 containing 2^31 - <ingress_correction_value> represented in binary.

47.3.9.8.2 Egress Correction

In the Transmit side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (Egress Correction Value) programed in the Egress Correction register. The value that needs to be programed in the egress correction register as mentioned below:

The timestamp correction because of synchronization is compensated by adding EGRESS_SYNC_CORR to the synchronized timestamp value.

When Enable one step timestamp feature is selected,

```
EGRESS_SYNC_CORR = (1*PTP_CLK_PER + 4*TX_CLK_PER)
```

Otherwise,

```
EGRESS_SYNC_CORR = -(2*PTP_CLK_PER)
```

The egress latency correction between the recommended capture point and the internal timestamp snapshot point is done by adding the latency value (EGRESS_LATENCY) with the captured timestamp.

```
Egress Correction = EGRESS_SYNC_CORR + EGRESS_LATENCY
```

Egress correction is performed by programming the TSEC field in the MAC Timestamp Egress correction register. The egress correction can be positive or negative and has a unit of nanoseconds. Negative values are represented in complement form.

When TSCTRLSSR bit in MAC_Timestamp_Control register is set, this has an accuracy of 1 ns.

If the correction is positive, it is represented by setting bit 0 to 31 and bits 1:31 containing <egress_correction_value> represented in binary. The value must not exceed 0x3B9A_C9FF. If the correction is negative, it is represented by setting bit 0 to 30 and bits 1:31 containing 10^9 - <egress_correction_value> represented in binary.

For example, if the required correction value is -5 ns, then the programed value should be 0xBB9A_C9FB

When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 ns. If the correction is positive, it is represented by setting bit 0 to 31 and bits 1:31 containing <egress_correction_value> represented in binary. The maximum value is 0x7FFF_FFFF. If the correction is negative, it is represented by setting bit 0 to 30 and bits 1:31 containing 2^31 - <egress_correction_value> represented in binary.

47.3.9.9 Frequency Range of Reference Timing Clock

The timestamp information is transferred across asynchronous clock domains, that is, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is 4 clock cycles of MII and 3 clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

47.3.9.9.1 Maximum PTP Clock Frequency

The maximum PTP clock frequency is limited by the maximum resolution of the reference time (1 ns resulting in 1 GHz) and the timing constraints achievable for logic operating on the PTP clock. In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

47.3.9.9.2 Minimum PTP Clock Frequency

The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time to the MII clock domain. This relationship is given in the following equation:

Equation 31

3 * PTP clock period + 4 * MII clock period <= Minimum gap between two SFDs

The MII clock frequency is fixed by IEEE specification. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in *Table 890*.

Note: It is recommended that you use a clock that has constant frequency, preferably a divided application clock.
When IEEE 1588 timestamp feature is enabled with internal timestamp, use a PTP clock frequency which is greater than 5 MHz. This is because the 8-bit MAC_Sub_Second_Increment register limits the minimum PTP frequency that can be used to ~4 MHz.

Table 890. Minimum PTP Clock Frequency Example

Mode	Minimum Gap Between Two SFDs	Minimum PTP Frequency with Internal Timestamp
10 Mbps full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
10 Mbps half duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz
100 Mbps full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
100 Mbps half duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz

47.3.9.10 PTP Processing and Control

Table 891 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008.

Table 891. Message Format Defined in IEEE 1588-2008

Bits								Octets	Offset
0	1	2	3	4	5	6	7		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField(1)								1	32
logMessageInterval								1	33

1. ControlField is used in version 1. In version 2, messageType field is used for detecting different message types.

There are some fields in the Ethernet payload that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP Packets over IPv4Packets over IPv4
- PTP Frames over IPv6Frames over IPv6
- PTP Packets over EthernetPackets over Ethernet

47.3.9.10.1 PTP Packets over IPv4

Table 892 provides information about the fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, *Annex D* and the message format defined in *Table 892*.

Table 892. IPv4-UDP PTP Packet Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Packet Type	12, 13	0x0800	IPv4 datagram
IP version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed: 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast Address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP Primary multicast address: 224.0.1.129 PTP Pdelay multicast address: 224.0.0.107
UDP Destination Port	36, 37	0x013F, 0x0140	0x013F: PTP event message[1] 0x0140: PTP general messages
PTP Control Field (IEEE 1588 version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP Message Type Field (IEEE 1588 version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	43 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

47.3.9.10.2 PTP Frames over IPv6

Table 893 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, *Annex D* and the message format defined in *Table 893*.

47.3.9.10.3 PTP Packets over Ethernet

Table 893. IPv6-UDP PTP Packet Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Packet Type	12, 13	0x86DD	IP datagram
IP Version	14 (Bits [0:3])	0x6	IP version is IPv6
Layer 4 Protocol	20(1)	0x11	UDP
PTP Multicast Address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP Primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP Destination Port	56, 57(1)	0x013F, 0x140	0x013F: PTP event message 0x0140: PTP general messages
PTP Control Field (IEEE 1588 version 1)	94(1)	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management (version1)
PTP Message Type Field (IEEE 1588 version 2)	62(1) (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The Extension Header is not defined for PTP packets.

Table 894 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, *Annex D* and the message format defined in Table 5-12 Message Format Defined in IEEE 1588-2008n 5-12.

47.3.9.11 One-Step Timestamp

Table 894. Ethernet PTP Packet Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Destination Multicast Address(1)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses(2): 01-1B-19-00-00-00 01-80-C2-00-00-0E(3)
MAC Packet Type	12, 13	0x88F7	PTP Ethernet packet
PTP Control Field (IEEE 1588 version 1)	46	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP Message Type Field (IEEE 1588 version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	15 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The unicast address match of destination addresses (DA), programed in MAC address 0 to 31, is used if the TSENMACADDR bit of MAC_Timestamp_Control register is set.
2. IEEE 1588-2008, *Annex F*
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

The Ethernet module supports the one-step timestamp feature. When this feature is enabled, the MAC identifies the offset in the packet and inserts the timestamp received from the application at that offset.

You can enable the one-step timestamp feature for a packet by setting Bit 11 (OSTC) in ATI Control Word. The inserted timestamp consists of 80-bits, the 64-bit TSSL and TSSH received from the application and the content of higher 16 bits of Higher Timestamp Word Register.

Note: The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets because the timestamp insertion happens in the MAC and the IP payload checksum needs to be updated. However, the checksum computation happens in the MTL.

47.3.9.11.1 MAC Transmit PTP Mode

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- Correction Field in the PTP header of messages
- OriginTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 895 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the *Section 47.2.140: Timestamp Control Register (MAC_TIMESTAMP_CONTROL)* and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 895. MAC Transmit PTP Mode and One-Step Timestamping Operation

Programming			Mode	Per Packet Control[1]			Messages Processed on Tx
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
X	X	X	N/A	1	X	X	Timestamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)
2'b00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
							Delay_Req (correction field for residence time and Egress Asym Cor)
2'b00	0	1	Ordinary or Boundary Slave	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for Egress Asym cor)
2'b00	1	1	Ordinary or Boundary Master	0	1	X	Sync (originTimestamp field) Sync (correction field for sub-nanosecond cor)
2'b01	X	0	End-to-End Transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
						Ingress TS	Pdelay_Req (correction field for residence time and Egress Asym Cor)
						Ingress TS	Pdelay_Resp (correction field for residence time and Ingress Asym Cor)

Table 895. MAC Transmit PTP Mode and One-Step Timestamping Operation (continued)

Programming			Mode	Per Packet Control[1]			Messages Processed on Tx
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
2’b01	0	1	Ordinary or Boundary Slave with support for peer delay mechanism or Peer to Peer Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor) (applicable only for Peer to Peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for Egress Asym cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)
2’b01	1	1	Ordinary or Boundary Master with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) Sync (correction field for sub-nanosecond cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)
2'b10	X	X	End-to-End Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
						Ingress TS	Delay_Req (correction field for residence time and Egress Asym Cor)
2'b11	X	X	Peer-to-Peer Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)

1. The per packet control values provided here are the recommended settings used by devices in typical PTP operation, for the programed mode.
2. TTSE represents TTSE bit of transmit descriptor in EQOS-AXI configuration, TTSE bit in TX control word in EQOS-MTL configuration, mti_ena_timestamp_i signal in EQOS-CORE configuration. The TTSE function is independent of the OST function and the programed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.
3. OSTC represents OSTC bit of transmit descriptor in EQOS-AXI configuration, OSTC bit in TX control word in EQOS-MTL configuration, mti_ost_en_i signal in EQOS-CORE configuration.

4. TTS represents the timestamp value provided in the TTSH, TTSL fields of transmit descriptor in EQOS-AXI configuration, timestamp provided in the TTSH, TTSL fields in the TX control word in EQOS-MTL configuration, mti_ost_i signal in EQOS-CORE configuration.
Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp.
When sub-nanosecond feature is enabled, residence time calculation includes sub-nanosecond accuracy.
Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

47.3.10 IPv4 ARP Offload

The Ethernet module supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. The Ethernet module generates the ARP reply packets for appropriate ARP request packets**.** The ARP packet for IPv4 is L2 layer packet with Length/Type of 0x0806.

The ARP offloading process is as follows:

1. The MAC receiver gets an ARP request if the Target Protocol Address of request matches the IPv4 address programed in the L3 register of the MAC
2. The MAC generates an ARP reply packet
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   – DA field of the Ethernet packet header
   – Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet
5. The MAC places its MAC address in the following fields:
   – SA field of the Ethernet packet header
   – Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply
8. The MAC recalculates the CRC and performs padding for generated ARP reply packet
9. The MAC transmitter sends the ARP reply

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to application with ARP Reply Not Generated (Bit 21) status bit set. However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request.

If the Disable CRC check bit of the MAC Extension Configuration bit is set, then the MAC does not check for valid CRC of an ARP request Packet. The ARP request Packet must always have a valid CRC. It can generate an ARP response packet if the other conditions are valid.

47.3.11 Memory Access in Debug Mode

The Ethernet module supports the debug access to the FIFO memory. When you enable this feature, you must program the entire selected Tx memory to Tx Queue 0 and Rx memory to Rx Queue 0. In addition, the offload features such as Checksum Offload Engine and IEEE timestamp should not be enabled with this feature.

When this feature is enabled, the MAC insert the pad (if required) and computed CRC in the Transmit packet. The FIFOs operate in the store-and-forward mode.

47.3.11.1 Slave Mode Access

You can use this mode to send or receive a packet through a slave interface. The following sections describe how to transmit or receive a packet through a slave interface.

47.3.11.1.1 Transmitting Packets through a Slave Interface

To write a packet to the Tx memory through a slave interface, complete the steps given in *Table 896*.

Table 896. Steps to Write a Packet to Tx memory through a Slave Interface

Register Name	Step Number and Description
MTL_DBG_CTL	Select the FIFO for slave access by using the DBGMOD and FDBGEN fields.
MTL_FIFO_DEBUG_DATA	Write the Packet control word.
MTL_DBG_CTL	Complete the following: a. Set the PKTSTATE field to Control field. b. Set the FIFOSEL field to Tx FIFO. c. Set the FIFOWREN bit. d. Reset the FIFORDEN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field to know the number of free locations.
MTL_FIFO_DEBUG_DATA	Write the SOP, the first word of the packet data.
MTL_DBG_CTL	Complete the following steps: a. Set the PKTSTATE field to SOP. b. Set the FIFOSEL field to Tx FIFO. c. Set the FIFOWREN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field to know the number of free locations.
MTL_FIFO_DEBUG_DATA	Write subsequent data words.
MTL_DBG_CTL	Complete the following steps: a. Set the PKTSTATE field to Packet Data. b. Set the FIFOSEL field to Tx FIFO. c. Set the FIFOWREN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required).
MTL_DBG_CTL and MTL_DBG_STS	Repeat Write subsequent data words. and Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required). till the penultimate word of the packet data.

Table 896. Steps to Write a Packet to Tx memory through a Slave Interface

Register Name	Step Number and Description
MTL_FIFO_DEBUG_DATA	Write the last data word in the FDBGDATA field.
MTL_DBG_CTL	Complete the following steps: a. Set the PKTSTATE field to EOP. b. Set the FIFOSEL field to Tx FIFO. c. Set the FIFOWREN bit. d. Set the number of valid bytes in the field.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field to know the number of free locations.

After Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field to know the number of free locations. in *Table 896*, the core automatically initiates the packet transmission. The status of the packet transmission can be obtained as explained in *Table 896*. To get the status of a Tx packet, complete the steps given in *Table 897*.

Table 897. Steps to Get the Status of a Tx Packet

Register Name	Step Number and Description
MTL_DBG_STS	Wait for the STSI interrupt bit to set.
MTL_DBG_CTL	Complete the following steps: a. Set the FIFOSEL field to Tx Status FIFO. b. Reset the FIFWREN bit. c. Set the FIFORDEN bit.
MTL_DBG_CTL	Read the FIFOBUSY bit and wait till this bit is low.
MTL_FIFO_DEBUG_DATA	Read the FDBGDATA field.

47.3.11.1.2 Receiving Packets through a Slave Interface

To read a packet from the Rx memory through a slave interface, complete the steps given in *Table 898*.

Table 898. Steps to Read a Packet from Rx memory through a Slave Interface

Register Name	Step Number and Description
MTL_DBG_CTL	Select the FIFO for slave access by using the DBGMOD and FDBGEN fields.
MTL_DBG_STS	Wait for the PKTI interrupt.
MTL_DBG_CTL	Complete the following: a. Set the FIFOSEL field to Rx FIFO. b. Reset the FIFOWREN bit. c. Set the FIFORDEN bit.

Table 898. Steps to Read a Packet from Rx memory through a Slave Interface

Register Name	Step Number and Description
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low.
MTL_DBG_CTL	Read the PKTSTATE and fields.
MTL_FIFO_DEBUG_DATA	Read the data in the FDBGDATA field.
MTL_DBG_STS	Read the LOCR field to know the number of available locations.
MTL_DBG_STS, MTL_DBG_CTL, MTL_FIFO_DEBUG_DATA	Repeat Read the FIFOBUSY bit and wait till this bit is low. through Read the LOCR field to know the number of available locations. till the complete packet, along with the Rx status, is received. **Note:** The format of the received packet is as follows First (One or multiple words) Last Status Word (One Word. Acts as delimiter between the status and data) Frame Data Word (multiple) Last Frame Data Word (One Word. Acts as a delimited among subsequent packets on the received Queue)

47.3.11.2 Debug Mode Access

Debug Mode enables you to do both the following operations:

- Write and Read back the Tx Queue
- Read and Write to the Tx Queue

This mode is useful for Memory BIST applications.

47.3.11.2.1 Writing to FIFO Memory in Debug Mode

To write to FIFO memory in the debug mode, complete the steps given in *Table 899*.

Table 899. Steps to Write to FIFO memory in Debug Mode

Register Name	Step Number and Description
MTL_DBG_CTL	Select the FIFO for debug access by using the DBGMOD and FDBGEN fields.
MTL_FIFO_DEBUG_DATA	Write the data in the FDBGDATA field.
MTL_DBG_CTL	Complete the following: – Set the FIFOSEL field to Tx FIFO, Rx FIFO, or TSO FIFO. – Reset the FIFORDEN bit. – Set the FIFOWREN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required).
MTL_DBG_CTL - MTL_DBG_STS	Repeat Complete the following: and Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required). twice to write one complete data word in 64-bit mode.

47.3.11.2.2 Reading from FIFO Memory in Debug Mode

To read from FIFO memory in the debug mode, complete the steps given in Table 900.

Table 900. Steps to Read from FIFO Memory in Debug Mode

Register Name	Step Number and Description
MTL_DBG_CTL	Select the FIFO for debug access by using the DBGMOD and FDBGEN fields. Complete the following: a. Set the FIFOSEL field to Tx FIFO, Rx FIFO, or TSO FIFO. b. Reset the FIFOWEN bit. c. Set the FIFORDEN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required).
MTL_FIFO_DEBUG_DATA	Read the data in the FDBGDATA field.
MTL_DBG_STS	Read the LOCR field (if required).
MTL_DBG_CTL, MTL_DBG_STS, MTL_FIFO_DEBUG_DATA	Repeat Complete the following: and Read the LOCR field (if required). twice to read one complete data word in 64-bit mode.

47.3.12 Loopback

The MAC supports Loopback of transmitted packets to its receiver. By default, the MAC Loopback function is disabled. You can enable this feature by programming the LM bit of the MAC_Configuration register.

You can enable loopback for all PHY interfaces. The data is always looped back on the MII interface irrespective of which PHY interface is selected. The loopback data is also passed through the corresponding interface block. The following are some guidelines for using the loopback mode:

- Enable loopback only with the full-duplex mode. In half-duplex mode, the carrier sense signal (crs) or collision (col) signal inputs get sampled which may result into issues such as packet dropping
- If the loopback mode is enabled without connecting a PHY chip, you should externally generate the Tx and Rx clocks and provide these clocks to the MAC
- Do not loop back big packets. Big packets may get corrupted in the loopback FIFO
- The MAC does not process ARP or PMT packets that are looped back

47.3.13 MAC Flow Control

This section describes the flow control for Transmit and Receive paths.

The Transmit Flow Control involves transmitting Pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

In the Receive path, the Flow Control is functional only in the full-duplex mode. The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

47.3.13.1 Transmit Flow Control

The Transmit Flow Control involves transmitting Pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

The Transmit Flow Control is enabled independently for each Rx queue when TFE bit is set in MAC_Q0_Tx_Flow_Ctrl /MAC_Q1_TX_Flow_Ctrl register.

SW can request the MAC to send a Pause packet or initiate backpressure by setting the FCB_BPA bit in the corresponding MAC_Q0_Tx_Flow_Ctrl /MAC_Q1_TX_Flow_Ctrl register.

HW can automatically trigger TX flow control based on the Rx Queue Threshold**.** The HW flow control operation of the MAC is enabled when the EHFC bit of the corresponding MTL_RxQ*n*_Operation_Mode register is set. The flow control signal to the MAC is asserted when the fill level of the Rx queue crosses the threshold configured in the RFA field of the MTL_RxQ*n*_Operation_Mode register. This flow control signal is de-asserted when the fill-level of the queue falls below the threshold configured in the RFD field.

Table 901, describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EHFC bit of the MTL_RxQ0_Operation_Mode register
- TFE bit of the MAC_Q0_Tx_Flow_Ctrl register
- DM bit of the MAC_Configuration register
- RFA field of the MTL_RxQ0_Operation_Mode register
- RFD field of the MTL_RxQ0_Operation_Mode register
- FCB_BPA bit of the MAC_Q0_Tx_Flow_Ctrl

Flow control is similar for Queue1.

Table 901. TX MAC Flow Control

EHFC	TFE	DM	Description
x	0	x	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs backpressure when FCB_BPA of the MAC_Q0_Tx_Flow_Ctrl register is set.
1	1	0	The MAC transmitter performs backpressure when FCB_BPA of the MAC_Q0_Tx_Flow_Ctrl register is set. In addition, the MAC TX performs backpressure when the Rx Queue level crosses the threshold set by the RFA field of the MTL_RxQ0_Operation_Mode register.
0	1	1	The MAC transmitter sends the pause packet when FCB_BPA of the MAC_Q0_Tx_Flow_Ctrl register is set.
1	1	1	The MAC transmitter sends the pause packet when FCB_BPA of MAC_Q0_Tx_Flow_Ctrl register is set. In addition, the MAC TX sends a pause packet when the Rx Queue level crosses the threshold set by the RFA field of the MTL_RxQ0_Operation_Mode register.

47.3.13.1.1 Flow Control in Full-Duplex Mode

In full-duplex mode, the Ethernet module uses IEEE 802.3x Pause packets for flow control. *Table 902*, describes the fields of a Pause packet. When the FCB_BPA bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Table 902. Pause Packet Fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets
PT	Contains Pause time specified in the PT field of the MAC_Q0_Tx_Flow_Ctrl /MAC_Q1_TX_Flow_Ctrl register

47.3.13.1.2 Flow Control in Half-Duplex Mode

In half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

47.3.13.2 Receive Flow Control

In the Receive path, the Flow Control is functional only in the full-duplex mode. If any Pause packet is received in the half-duplex mode, the packet is considered as a normal control packet. You can enable the Pause flow control by setting the RFE bit in the MAC_Rx_Flow_Ctrl register. *Table 903*, describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of MAC_Rx_Flow_Ctrl register
- DM bit of MAC_Configuration register

Table 903. RX MAC Flow Control

RFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.
1	1	The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

When the RFE bit is enabled, the MAC transmitter is blocked when the 802.3x Pause packet is received.

The following list describes the Rx flow control:

1. The MAC checks the destination address of the received Pause packet for either of the following:
   - Multicast destination address: The DA matches the unique multicast address specified for the control packet (48'h0180C2000001)
   - Unicast destination address: The DA matches the content of the MAC Address Register 0 and the UP bit of MAC_Rx_Flow_Ctrl register is set.
     If the UP bit is set and the MAC processes Pause packets with unicast destination address in addition to the unique multicast address
2. The MAC decodes the following fields of the received packet:
   - Type field: This field is checked for 16'h8808
   - Opcode field: This field is checked for 16'h0001 (Pause packet)
   - Pause Time: The Pause time (for Pause packet) is captured to determine the time for which transmitter needs to be blocked. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times)
3. The MAC transfers the received control packet to the application based on the setting of the PCF field in MAC_Packet_Filter register
4. If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value

47.3.14 Checksum Offload Engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. Therefore, the MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, and error detection in the Receive path. On the transmit path the Checksum Offload Engine is present in Queue0 only.

47.3.14.1 Transmit Checksum Offload Engine

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 Bits[14:15]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, when this function is enabled, the Tx FIFO automatically operates in the store-and-forward mode even if the Ethernet module is configured for Threshold (cut-through) mode.
You must make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter. The reason being that when space is not available to accept the programed burst length of data, then the MTL Tx FIFO starts reading to avoid dead-lock. When reading starts, the COE fails and consequently all succeeding packets may get corrupted because of improper recovery. Therefore, you must enable the checksum insertion only in the packets that are less than the following number of bytes in size: TXFIFO_SIZE – ((PBL + N)(DATAWIDTH/8)), Where N= 5*

47.3.14.1.1 IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 31). This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes)
  - The total packet length is less than the value given in the IPv4 Header Length field
- For IPv6 datagrams:
  - The Ethernet type is 0x86dd but the IP header Version field is not equal to 0x6
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received

47.3.14.1.2 TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16'h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (Bit 19). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC Transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 904 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Note: You should not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Transmit Checksum Offload Engine because it may result in incorrect checksum insertion or unexpected behavior.

Table 904. Transmit Checksum Offload Engine Functions for Different Packet Types

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 packet	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine.	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2 K Packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine.	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv6 packet with the following next header fields in main or extension headers		
Hop-by-hop options (in IPv6 main header)	Not applicable	Yes
Hop-by-hop options (in IPv6 extension header)	Not applicable	No
Destinations options	Not applicable	Yes
Routing (with segment left 0)	Not applicable	No
Routing (with segment left > 0)	Not applicable	No
TCP, UDP, or ICMP	Not applicable	Yes
Authentication	Not applicable	Yes
Any other next header field in main or extension headers	Not applicable	No
IPv4 Tunnels		
IPv4 packet in an IPv4 tunnel	Yes (IPv4 tunnel header)	No
IPv6 packet in an IPv4 tunnel	Yes (IPv4 tunnel header)	No
IPv6 Tunnels		
IPv4 packet in an IPv6 tunnel	Not applicable	No
IPv6 packet in an IPv6 tunnel	Not applicable	No

Table 904. Transmit Checksum Offload Engine Functions for Different Packet Types (continued)

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

47.3.14.2 Receive Checksum Offload Engine

You can enable the Receive Checksum Offload Engine (Rx COE) by setting the IPC bit of MAC_Configuration register. When enabled, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the MAC_VLAN_Tag register is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is sent in the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 905, describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status*Table 906*.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 905. Receive Checksum Offload Engine Functions for Different Packet Types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers		
Hop-by-hop options (in IPv6 main header)	Not applicable	Yes
Hop-by-hop options (in IPv6 extension header)	Not applicable	No
Destinations options	Not applicable	Yes
Routing (with segment left 0)	Not applicable	Yes
Routing (with segment left > 0)	Not applicable	No
TCP, UDP, or ICMP	Not applicable	Yes
Any other next header field in main or extension headers	Not applicable	No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels		
IPv4 packet in an IPv4 tunnel	Yes (IPv4 tunnel header)	No
IPv6 packet in an IPv4 tunnel	Yes (IPv4 tunnel header)	No
IPv6 Tunnels		
IPv4 packet in an IPv6 tunnel	Not applicable	No
IPv6 packet in an IPv6 tunnel	Not applicable	No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

47.3.15 MAC Management Counters

The counters in the MAC Management Counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the CSR. Each register is 32-bits wide. The width of each MMC counter is 32-bit.

The MMC counters are free running. There is no separate enable for the counters to start. If a particular MMC counter is present, it starts counting when corresponding packet is received or transmitted. The Receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets, dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, you should set Bit 31 in the MAC_Packet_Filter register.

The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

The following definitions define the terminology.

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber Timeout
  - No Carrier or Loss of Carrier
  - Late Collision
  - Packet Underflow
  - Excessive Deferral
  - Excessive Collision
- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbps only)
  - Length error (non-Type packet only)
  - Out of Range (non-Type packet only, longer than 1518 bytes)
  - MII_RXER Input error
- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN Frame maxsize = 1,522
  - Jumbo Frame maxsize = 9,018
  - JumboVLAN Frame maxsize = 9,022
- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in *Table 906*.

Table 906. Size of the Maximum Receive Packet Size

JE	S2KP	GPSLCE	EDVLP	Untagged Frame maxsize in bytes	Single VLAN Frame maxsize in bytes	Double VLAN Frame maxsize in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

47.3.16 Low Power Modes

This chapter describes the low power modes supported by the Ethernet module. It contains the following sections:

- Energy Efficient Ethernet
- Power Management

47.3.16.1 Energy Efficient Ethernet

Energy Efficient Ethernet (EEE) is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of Physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps. The Ethernet module supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows power saving by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities, negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Note: The EEE feature is not supported when the MAC is configured to use the RMII PHY interface. Even if the MAC supports multiple PHY interfaces, you should activate the EEE mode only when the MAC is operating with the MII interface.
According to the Energy Efficient Ethernet standard (IEEE 802.3az-2010), the LPI mode is supported only in the full-duplex mode. Therefore, you should not enable the LPI mode when the MAC Transmitter is configured for the half-duplex mode.

47.3.16.1.1 Transmit Path Functions

In the Transmit path, the software must set the LPIEN bit of the MAC_LPI_Control_Status register to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER field of MAC_LPI_Timers_Control register. The PHY Link Status bit of the LPI Control and Status Register indicates the link status of the PHY.

Note: According to the Energy Efficient Ethernet standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN
2. Asserts TX_ER
3. Sets TXD[3:0] to 0x1 (for 100 Mbps)
4. Updates the status (TLPIEN bit of MAC_LPI_Control_Status register) and generates an interrupt

To bring the PHY out of the LPI state, that is, when the software resets the LPIEN bit, the MAC performs the following tasks:

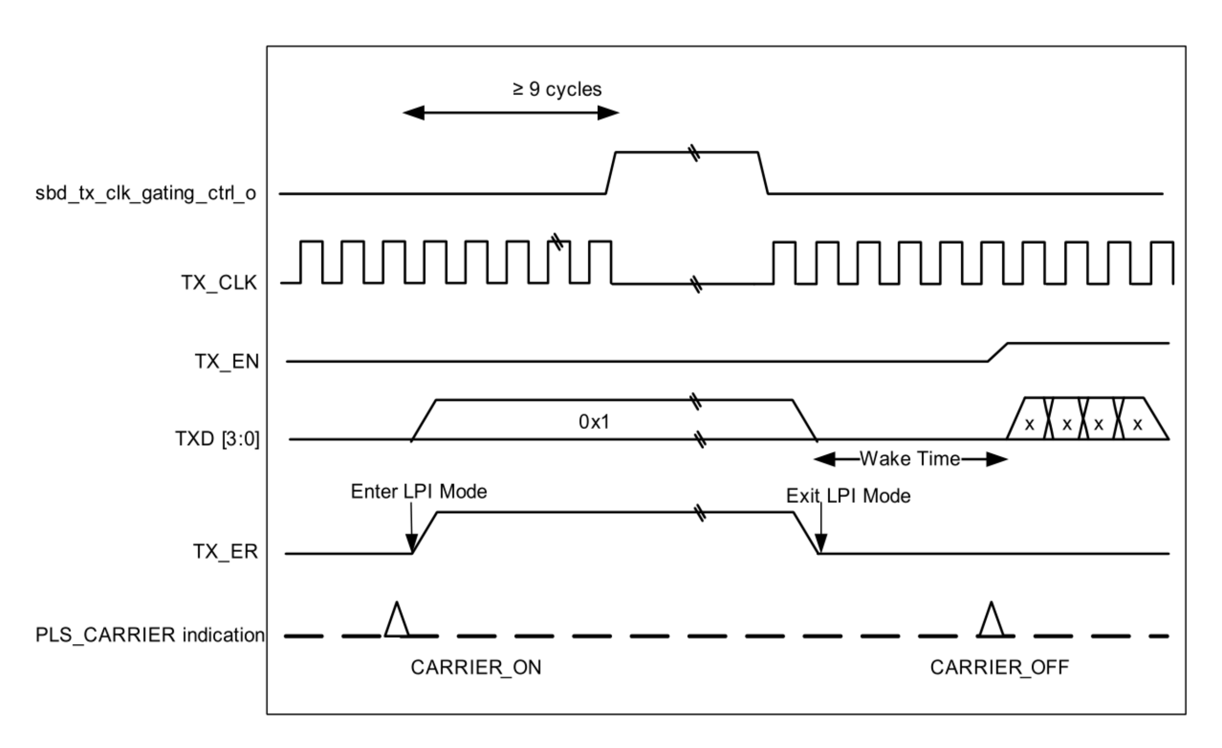
1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern
2. Starts the LPI TW TIMER
   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programed in the TWT field of the MAC_LPI_Timers_Control register
3. Updates the LPI exit status (TLPIEX bit of the MAC_LPI_Control_Status register) and generates an interrupt

Figure 826 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

The MAC does not stop the TX_CLK clock. You can stop this clock (as shown in Figure 826) if your PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o signal to 1.
The sbd_tx_clk_gating_ctrl_o signal is asserted after 9 Tx Clock Cycles, one Pulse Synchronizer delay (BCM22), and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal is dependent on the LPITCSE bit of the MAC_LPI_Control_Status Register.
If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.
If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

Figure 826. LPI Transitions (Transmit)



47.3.16.1.1.1 Automated Entry/Exit of LPI mode in TX Path

The MAC transmitter can be programed to enter and exit LPI IDLE mode automatically based on whether it is IDLE for a specific period of time or has a packet to transfer. These modes are enabled and controlled by MAC_LPI_Control_Status register.

When LPITXA (Bit[12]) and LPITXEN (Bit[15]) of MAC_LPI_Control_Status register are set, the MAC transmitter enters LPI IDLE state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter will exit the LPI IDLE state and clear the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition to the above, when Bit[11] (LPIATE) is also set, the MAC transmitter will enter LPI IDLE state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in MAC_LPI_Entry_Timer. In this mode also, the MAC transmitter will exit the LPI IDLE state as soon as any of the functions becomes non-idle. However, the LPITXEN bit is not cleared but remains active so that re-entry to LPI IDLE state is possible without any software intervention when the MAC becomes idle again.

When both LPIATE and LPITXA bits are cleared, you can directly control the entry and exit of LPI IDLE state by programming the LPITXEN bit.

47.3.16.1.2 Receive Path Functions

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER
2. The PHY sets RXD[7:0] to 0x01
3. The PHY de-asserts RX_DV
4. The MAC updates the RLPIEN bit of the MAC_LPI_Control_Status register and immediately generates an interrupt

Note: If the LPI pattern is detected for a very short duration (that is, less than 2 cycles of Rx clock), the MAC does not enter the Rx LPI mode.
If the duration between end of the current Rx LPI pattern and start of the next Rx LPI pattern, is very short (that is, less than 2 cycles of Rx clock), then the MAC exits and again enters the Rx LPI mode. The MAC does not give the Rx LPI Exit and Entry interrupts.

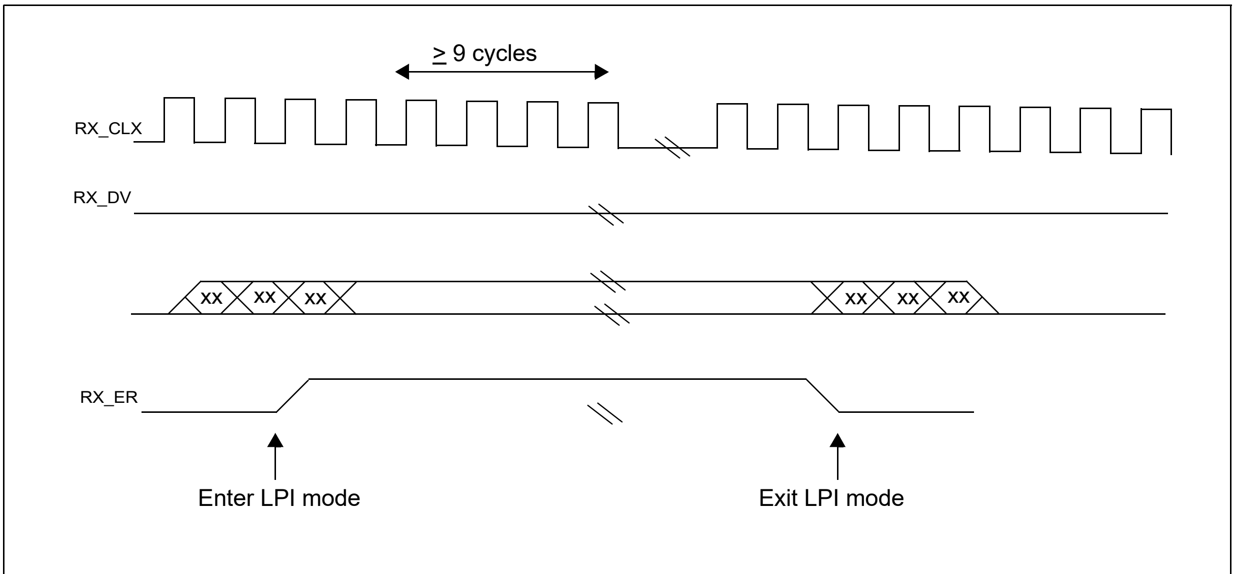
The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the MAC_LPI_Control_Status register and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 827 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 827. LPI Transitions (Receive)



Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than 9 clock cycles after the start of the LPI state as shown in Figure 827.
If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC receiver comes out of the LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver again enters the LPI state.
If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.
When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically

enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.

47.3.16.1.3 LPI Timers

The transmitter maintains the following two timers that are loaded with the respective values from the MAC_LPI_Timers_Control and MAC_LPI_Entry_Timer registers:

- LPI LS TIMER: The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. Software should get the PHY link status by reading the PHY register and accordingly update the PLS bit.
  This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programed in Bits[6:15] of the MAC_LPI_Timers_Control register. The MII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. Therefore, the software can program up to 1023 milliseconds.
- LPI TW TIMER: The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programed in Bit[16:31] of MAC_LPI_Timers_Control register. The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count.
  The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 µs.
- LPI AUTO ENTRY TIMER: This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC Transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in MAC_LPI_Control_Status register is set.

47.3.16.2 Power Management

The power management (PMT) block supports the reception of network (remote) wake-up packets and magic packets. The PMT block does not perform the clock gate function, but generates interrupts for remote wake-up packets and magic packets that the MAC receives.

When you enable the power-down mode in the PMT block, the MAC drops all received packets and does not forward any packet to the MTL RxFIFO or the application. The MAC comes out of the power-down mode only when a magic packet or a remote wake-up packet is received and the corresponding detection is enabled.

The PMT block is available in the receive path of MAC. You can select both types of power management packet (remote wake-up packet and magic packet). You can use the RWKPKTEN and MGKPKTEN bits of the MAC_PMT_Control_Status register to generate power management events. The application should program these bits. You can access the PMT registers in the similar manner as you access the MAC CSR registers.

Note: The magic packet feature is implemented based on the Magic Packet Technology white paper. The remote wake-up packet feature is implemented based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

47.3.16.2.1 PMT Block Registers

The following are the PMT block registers:

- MAC_PMT_Control_Status
- MAC_RWK_Packet_Filter

47.3.16.2.2 Remote Wake-Up Packet Detection

When the MAC is in sleep mode and the remote wake-up bit is enabled in the MAC_PMT_Control_Status, the normal operation is resumed after a remote wake-up packet is received. The application writes all eight wake-up filter registers, by performing a sequential Write to address (00C4H). The application enables remote wake-up by writing 1 to the RWKPKTEN bit of the MAC_PMT_Control_Status register.

The PMT block supports 4 programmable filters that allow support of different receive packet patterns. If the incoming packet passes the address filtering of Filter Command, and if Filter CRC-16 matches the CRC of the incoming pattern, the MAC identifies the packet as a wake-up packet.

The Filter Offset determines the offset from which the packet is to be examined. The Filter Byte Mask determines which bytes of the packet must be examined. The bit 0 of Byte Mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The remote wake-up packet is checked only for length error, FCS error, dribble bit error, MII error, and collision. In addition, the remote wake-up packet is checked to ensure that it is not a runt packet. Even if the remote wake-up packet is more than 512 bytes long, if the packet has a valid CRC value, it is considered valid. The remote wake-up packet detection is updated in the PMT Control and Status register for every remote wake-up packet received. A PMT interrupt to the application triggers a Read to the PMT Control and Status register to determine reception of a remote wake-up packet.

Note: The watchdog timeout limit for a remote wake-up frame is 2,048 bytes irrespective of the value programed in WD bit of MAC_Configuration register and PWE bit in MAC_Watchdog_Timeout register.
The value programed in DCRCC bit of MAC_Ext_Configuration register is applicable to a remote wake-up frame only when Enable IPv4 ARP Offload is selected in the configuration.

47.3.16.2.3 Magic Packet Detection

The magic packet is based on a method that uses the magic packet technology from Advanced Micro Device to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network.

The MAC checks only those magic packets that are addressed to the MAC or a multicast address (including broadcast address) to determine whether these packets meet the wake-up requirements. The magic packets that pass the address filtering (unicast or multicast (including broadcast) address) are checked to determine whether they meet the remote wake-up packet data format of 6 bytes of all ones followed by a Unicast MAC Address (that matches the value in MAC Address 0) appearing 16 times.

The application enables the magic packet wake-up by writing 1 to the MGKPKTEN bit of the MAC_PMT_Control_Status register. The PMT block constantly monitors each packet addressed to the node for a specific magic packet pattern. Each packet received is checked for a 48’hFF_FF_FF_FF_FF_FF pattern following the destination and source address field.

The PMT block then checks the packet for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the PMT block again scans the 48’hFF_FF_FF_FF_FF_FF pattern in the incoming packet. The 16 repetitions can be anywhere in the packet, but must be preceded by the synchronization stream (48’hFF_FF_FF_FF_FF_FF). The device can also accept a multicast packet, as long as the 16 duplications of the MAC address are detected. If the number of repetitions of 8'hFF are more than 6, the PMT block checks for 16 repetitions of the MAC address without any breaks or interruptions, after the last 6 repetitions of 8'hFF.

If the MAC address of a node is 48'h00_11_22_33_44_55, the MAC scans for the following data sequence:

Destination Address Source Address ……………….. FF FF FF FF FF FF

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

…CRC

The Ethernet module checks the remote wake-up packet only for length error, FCS error, dribble bit error, MII error, collision. In addition, the remote wake-up packet is checked to ensure that it is not a runt packet. Even if the remote wake-up packet is more than 512 bytes long, if the packet has a valid CRC value, the Ethernet module considers it a valid packet.

The magic packet detection is updated in the MAC_PMT_Control_Status register for the received magic packet. A PMT interrupt to the Application triggers a read to the MAC_PMT_Control_Status register to determine whether a magic packet has been received.

Note: The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programed in WD bit of MAC_Configuration register and PWE bit in MAC_Watchdog_Timeout register.
The value programed in DCRCC bit of MAC_Ext_Configuration register is applicable to a magic packet only when Enable IPv4 ARP Offload is selected in the configuration.

47.3.16.2.4 System Considerations during Power Down

The Ethernet module neither gates nor stops clocks when the power-down mode is enabled. Power saving by clock gating must be done outside the core by the application. The receive data path must be clocked with clk_rx_i during the power-down mode, because it is involved in the magic packet or remote wake-up packet detection. However, the application path clock can be gated during the power-down mode. The transmit Path clock must be gated during the power-down mode only if ARP Offload is not enabled.

The recommended power-down and wake-up sequence is as follows.

1. Disable the Transmit DMA (if applicable) and wait for any previous packet transmissions to complete. These transmissions can be detected when Transmit Interrupt [Bit 31 of the register] is received
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the_ MAC_Configuration register
3. Wait until the Receive DMA empties all the packets from the Rx FIFO to system memory. You can do this by reading the appropriate bits of Debug registers in the DMA and MTL CSR space
4. Enable Power-Down mode by appropriately configuring the PMT registers
5. Enable the MAC Receiver and enter the Power-Down mode
6. Gate the application and transmit clock (transmit clock must be gated only if ARP Offload is not enabled) inputs to the core (and other relevant clocks in the system) to reduce power and enter Sleep mode
7. On receiving a valid remote wake-up packet, the Ethernet module asserts the pmt_intr_o signal and exits the Power-Down mode
8. On receiving the interrupt, the system must enable the application and transmit clock inputs to the core
9. Read the PMT Status register to clear the interrupt, then enable the other modules in the system and resume normal operation

47.3.17 Interrupts

Interrupt can be generated as a result of various events in the Ethernet IP core. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal (interrupt) is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the DMA_Interrupt_Status register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- DMA_CH*n*_Status register
- MTL_Interrupt_Status register
- MAC_Interrupt_Status register

The interrupt is a level signal and will get de-asserted only when all the enabled interrupt events are cleared in their respective status registers and correspondingly all the bits in the DMA_Interrupt_Status register are cleared.

47.3.17.1 DMA Interrupts

The DMA_CH*n*_Status register captures all the interrupt events of that TxDMA and RxDMA channel pair. The DMA_CH*n*_Interrupt_Enable register contains the corresponding enable bits for each of the interrupt event. There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts. They are indicated by Bits[16:17] of DMA_CH*n*_Status register respectively. The normal group is for events that happen during the normal transfer of packets (TI, RI, TBU) while the abnormal interrupt events are for error events. Interrupt events are cleared by writing 1 to the corresponding bit position. When all the enabled interrupt events are cleared (including the NIS and AIS), the interrupt source from the DMA Channel is cleared and the corresponding bit in register is also cleared.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. For example, Receive Interrupt Bit[25] of DMA_CH*n*_Status register indicates that one or more packets were transferred to the application buffer. The driver must scan all descriptors, from the last recorded position to the first one, owned by the DMA to determine how many packets are received.

An interrupt is generated only once for multiple events. The driver must scan the register for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of the DMA_Interrupt_Status register are cleared.

47.3.17.1.1 Periodic scheduling of DMA Transmit and Receive Interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet IP gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[0]) once for every “required” number of packets to be transmitted
2. Similarly, set the IOC (RDES3[1]) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated

In addition to above, an interrupt timer (DMA_CH*n*_Rx_Interrupt_Watchdog_Timer) is given for flexible control and periodic scheduling of Receive Interrupt. When this interrupt timer is programed with a non-zero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive Interrupt because the corresponding interrupt of completion IOC bit (RDES3[1]) is not set. When this timer runs out as per the programed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in DMA_CH*n*_Interrupt_Enable register. The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor’s IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

47.3.17.1.2 Per DMA channel Transfer Complete Interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in DMA_CH*n*_Status register. The TI bit is set whenever the Tx DMA channel closes the a descriptor in which the IOC (Interrupt On Completion - TDES2[0]) bit is set. Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with LD bit set and in any of the descriptors used for transferring that packet, IOC (Interrupt Enable on completion - RDES3[1]) bit is set.

The common interrupt output signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in DMA_CH*n*_Interrupt_Enable register.

47.3.17.2 MTL Interrupts

In MTL, the interrupts are mainly related to exception events in the TxQ or RxQ in the Transmit and Receive paths respectively. The interrupt status are captured and organized in a hierarchical manner in order to identify the root cause quickly.

The MTL_Interrupt_Status register identifies the top level modules that can cause the interrupt to be asserted.

- Bits[30:31] identify 2 Queues (Tx or Rx). They are read-only bits and the application should read the corresponding MTL_Q*n*_Interrupt_Status register to identify the exact cause and set the corresponding bits to 1 to clear the interrupt event. The assertion of interrupt due to these events is enabled by the corresponding enable bits in MTL_Q*n*_Interrupt_Enable register
- Bit[15] is made high in case the event is present in the MAC core. The application should in turn read the MTL_Interrupt_Status register to identify the cause
- Bit[14] is related to indirect access completion events to the TxFIFO and RxFIFO memory in Debug Mode

47.3.17.3 MAC Interrupts

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The MAC_Interrupt_Status register describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the MAC_Interrupt_Enable register.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt. For example, when set high, Bit 26 of the MAC_Interrupt_Status register indicates that the LPI interrupt status has changed. You must read the TLPIEN bit of MAC_LPI_Control_Status register to clear this interrupt event.

The interrupts from the MAC are combined (OR'ed) and given as the MACIS bit in the DMA_Interrupt_Status register.

47.3.17.4 LPI Interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt mci_intr_o (interrupt in EQOS-DMA configurations) is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the MAC_LPI_Control_Status register.

When the MAC exits the Rx LPI state, then in addition to the mci_intr_o (interrupt in EQOS-DMA configurations), the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. You can use the lpi_intr_o signal to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that you can stop the application clock when the MAC is in the LPI state. If you do not want to gate-off the application clock during the Rx LPI state, you can leave the lpi_intr_o signal unconnected and use the mci_intr_o (interrupt in EQOS-DMA configurations) signal to detect Rx LPI exit.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the MAC_LPI_Control_Status register is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the Ethernet module is operating in the 10 Mbps mode.

47.3.17.5 PMT Interrupts

The PMT interrupt signal is asserted when a valid remote wake-up packet is received. In addition to the interrupt, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that you can stop the application clock when the MAC is in the power-down mode.

You can use the pmt_intr_o interrupt to trigger the external clock-gating circuitry to restore the application clock and Tx clock to the MAC. If you do not want to gate-off these clocks during the power-down mode, you can leave the pmt_intr_o signal unconnected and use the interrupt to detect the MAC wake-up operation.

The pmt_intr_o signal is generated in the Rx clock domain. It is not cleared immediately when the PMT Control and Status register is read. This is because the resultant clear signal has to cross to the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the Ethernet module is operating in the 10 Mbps mode.

When software resets the PWRDWN bit in Remote Wake-Up Packet Detection register, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

47.4 Descriptors

47.4.1 Overview

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory. The Ethernet module supports the following two types of descriptors:

- Normal Descriptor: Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted
- Context Descriptor: Context descriptors are used to provide control information applicable to the packet to be transmitted

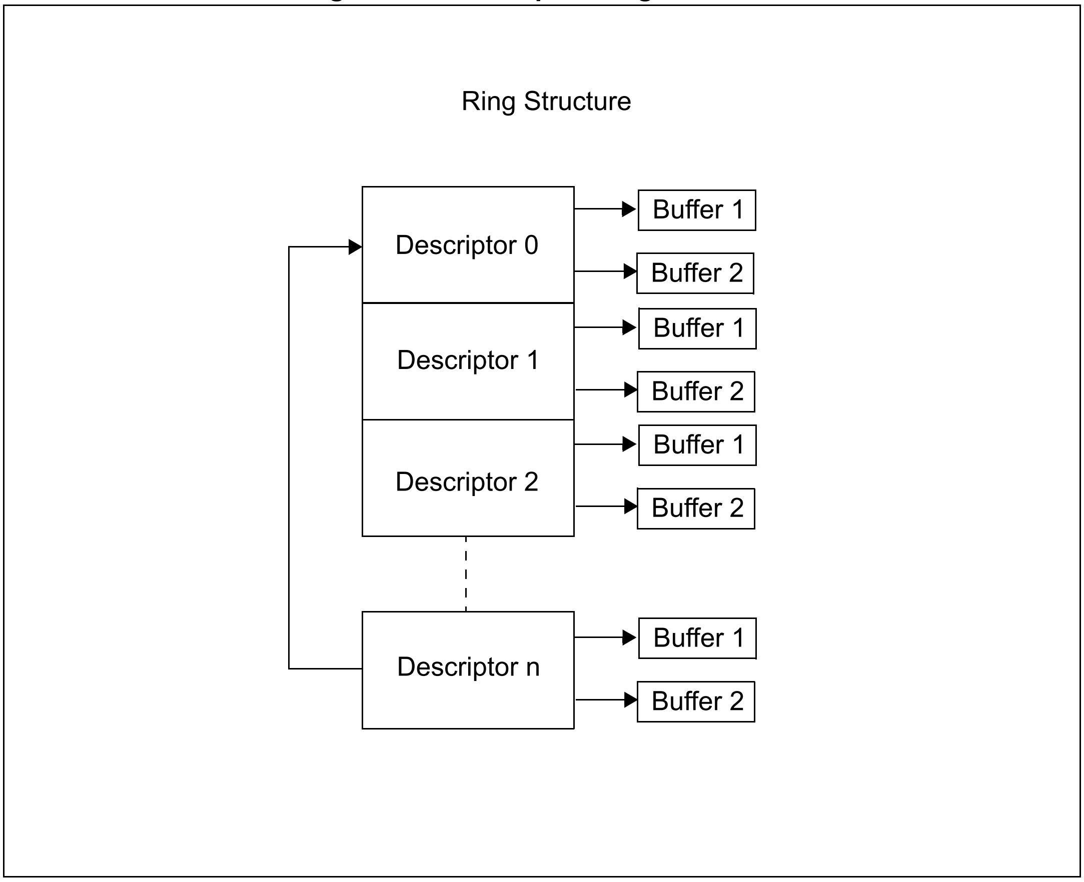
Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

Note: There is no limit for number of descriptors that can be used for a single packet.

47.4.1.1 Descriptor Structure

The Ethernet module supports the ring structure for DMA descriptor.

Figure 828. Descriptor Ring Structure



In Ring structure, descriptors are separated by the DWord number programed in the DSL field of the DMA_CH*n*_Control register. The application needs to program the total ring length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

- Transmit Descriptor Ring Length Register (DMA_CH*n*_TxDesc_Ring_Length)
- Receive Descriptor Ring Length Register (DMA_CH*n*_RxDesc_Ring_Length)

The Descriptor Tail Pointer Register contains the pointer to the descriptor address (*N*). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (*N* – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

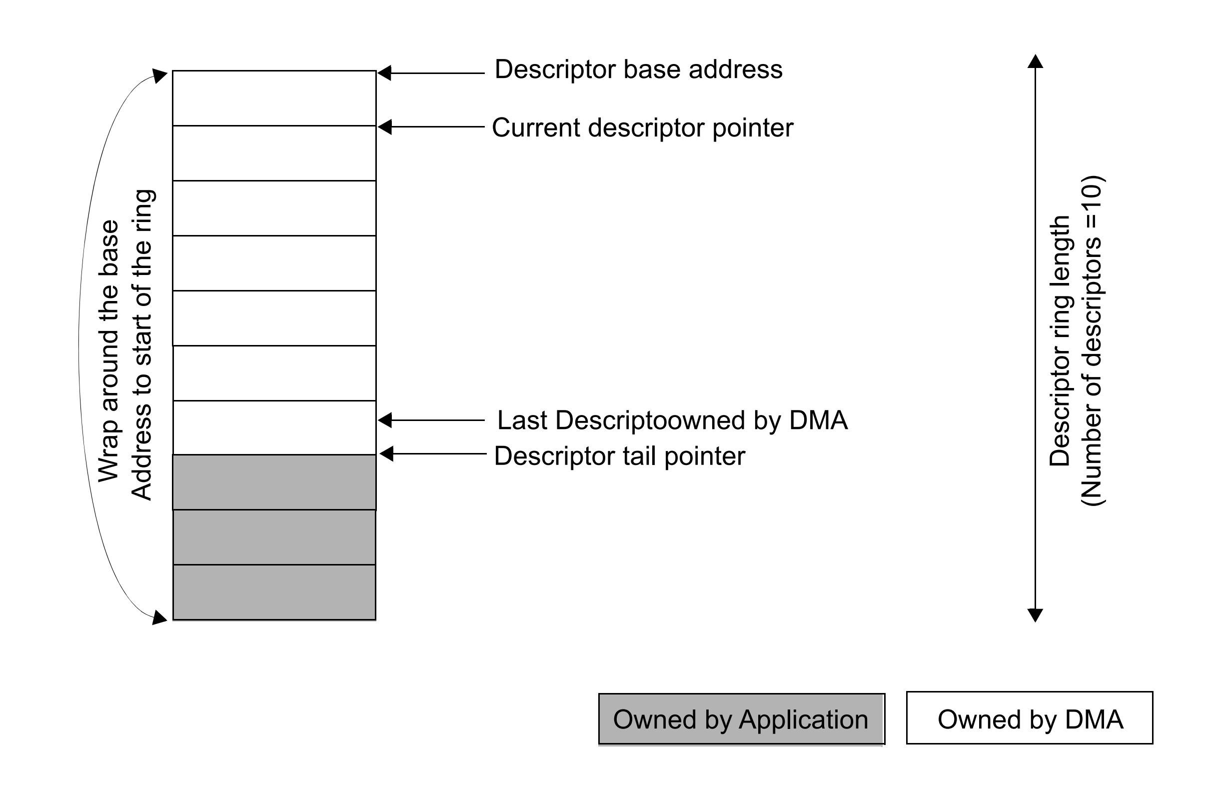
```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA goes into the Suspend mode when this condition occurs. The application must perform a write to the Descriptor Tail pointer register and update the tail pointer so that the following condition is true:

```
Current Descriptor Pointer < Descriptor Tail Pointer;
```

The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 829*.

Figure 829. DMA Descriptor Ring



For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

47.4.1.2 Descriptor Endianness

The descriptor addresses must be aligned to configured bus width (DWord for 64-bit bus). The data bus can be configured for either little-endian or big-endian format.

Figure 830 shows the normal Receive and Transmit descriptors for 64-bit data bus when the endian mode of data bus is little-endian or big-endian and the endian mode of descriptors is reverse-endian.

Figure 830. Rx/Tx Descriptors in Same-Endian Mode for 64-Bit, Little-Endian Data Bus

63	31 0
DES1	DES0
DES3	DES2

Figure 830 and *Figure 831* show the normal Receive and Transmit descriptors for 64-bit data bus when the endian mode of data bus is big-endian and the endian mode of descriptors is reverse-endian. In 64-bit big-endian data bus systems, when the descriptors are configured for reverse endianness, the order of the descriptor subset is reversed from the default (7-3 Rx/Tx Descriptors in Same-Endian Mode for 64-Bit, Little-Endian Data Bus). For example, Bits[63:32] are DES0 while Bits[31:0] are DES1.

Figure 831. Rx/Tx Descriptors in Reverse-Endian Mode for 64-Bit, Big-Endian Data Bus

63	31 0
DES0	DES1
DES2	DES3

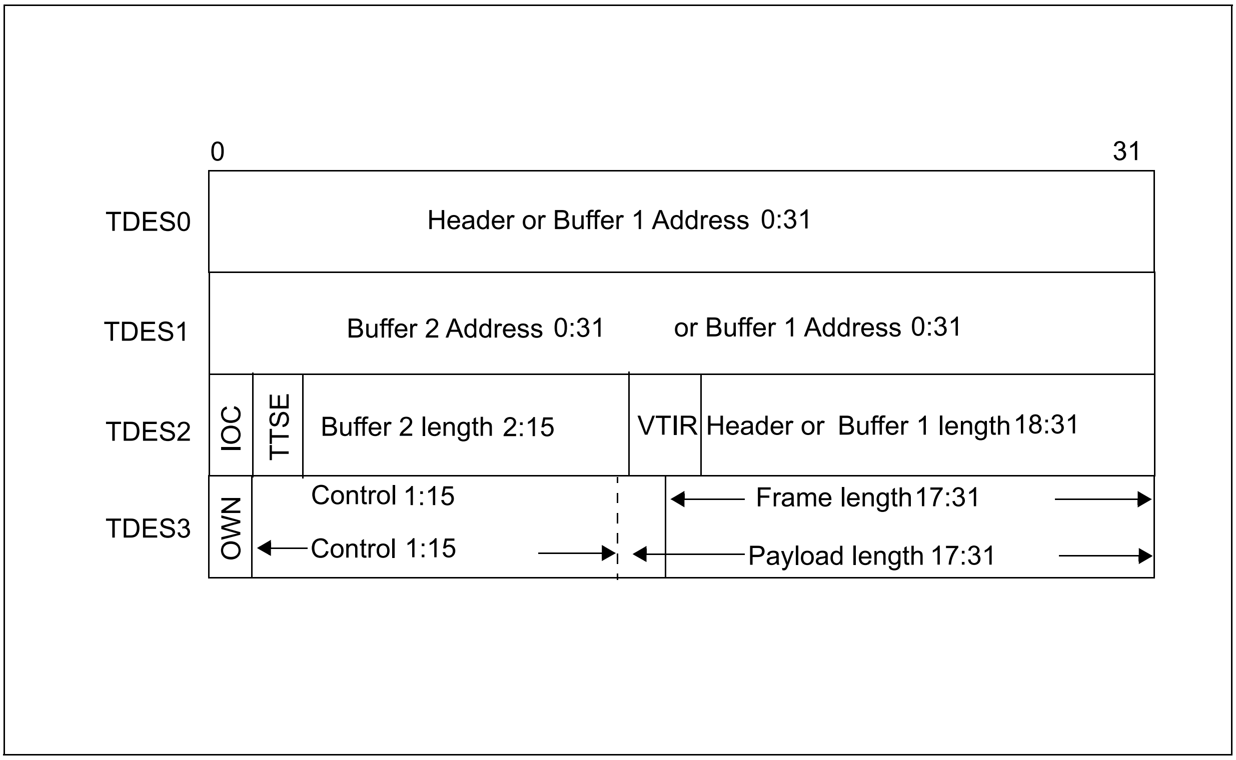
47.4.1.3 Transmit Descriptor

The DMA in the Ethernet module requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit packet basis. The Transmit Normal descriptor has the following two formats: Read format and Write-Back format

47.4.1.3.1 Transmit Normal Descriptor (Read Format)

Figure 832 shows the Read Format for a Transmit normal descriptor.

Figure 832. Transmit Descriptor Read Format



47.4.1.3.1.1 TDES0 Normal Descriptor (Read Format)

Table 907. TDES0 Normal Descriptor (Read Format)

Field	Description
0:31 BUF1AP	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1.

47.4.1.3.1.2 TDES1 Normal Descriptor (Read Format)

Table 908. TDES1 Normal Descriptor (Read Format)

Field	Description
0:31 BUF2AP	Buffer 2 Address Pointer This bit indicates the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment.

47.4.1.3.1.3 TDES2 Normal Descriptor (Read Format)

Figure 833. TDES2 Normal Descriptor (Read Format)

0	1	2:15	16:17	18:31
IOC	TTSE	B2L	VTIR	B1L

Table 909. TDES2 Normal Descriptor (Read Format)

Field	Description
0 IOC	Interrupt on Completion This bit sets the TI bit in the DMA_CH*n*_Status register after the present packet has been transmitted.
1 TTSE	Transmit Timestamp Enable This bit enables the IEEE1588 time stamping for Transmit packet referenced by the descriptor.
2:15 B2L	Buffer 2 Length The driver sets this field. When set, this field indicates Buffer 2 length.
16:17 VTIR	VLAN Tag Insertion or Replacement These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN Tag Insertion, Replacement, or Deletion is enabled for the packet. The following list describes the values of these bits: 00 Do not add a VLAN tag. 01 Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. 10 Insert a VLAN tag with the tag value programed in the MAC_VLAN_Incl register or context descriptor. 11 Replace the VLAN tag in packets with the tag value programed in the MAC_VLAN_Incl register or context descriptor. This option should be used only with the VLAN packets.
18:31 B1L	Buffer 1 Length

47.4.1.3.1.4 TDES3 Normal Descriptor (Read Format)

Figure 834. TDES3 Normal Descriptor (Read Format)

0	1	2	3	4:5	6:8	9:12	13	14:15	16	17:31
OWN	CTXT	FD	LD	CPC	SAIC	SLOTNUM	Rsvd	CIC	Reserved	FL

Table 910. TDES3 Normal Descriptor (Read Format)

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
1 CTXT	Context Type This bit should be set to 1'b0 for normal descriptor.
2 FD	First Descriptor When this bit is set, it indicates that the buffer contains the first segment of a packet.

Table 910. TDES3 Normal Descriptor (Read Format) (continued)

Field	Description
3 LD	Last Descriptor When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value.
4:5 CPC	CRC Pad Control This field controls the CRC and Pad Insertion for Tx packet. This field is valid only when the first descriptor bit (TDES3[2]) is set. The following list describes the values of Bits[4:5]: 00 CRC and Pad Insertion - The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 01 CRC Insertion (Disable Pad Insertion) - The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit Buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. 10 Disable CRC Insertion - The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. 11 CRC Replacement - The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer.
6:8 SAIC	SA Insertion Control These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must set the CRC Pad Control bits appropriately when SA Insertion Control is enabled for the packet. Bit 6 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[7:8]: 00 Do not include the source address 01 Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 10 Replace the source address. For reliable transmission, the application must provide frames with source addresses. 11 Reserved
9:12 SLOTNUM	SLOTNUM: Slot Number Control Bits in AV Mode These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the Transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field DMA_CH*n*_Slot_Function_Control_Status. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels.
13	Reserved

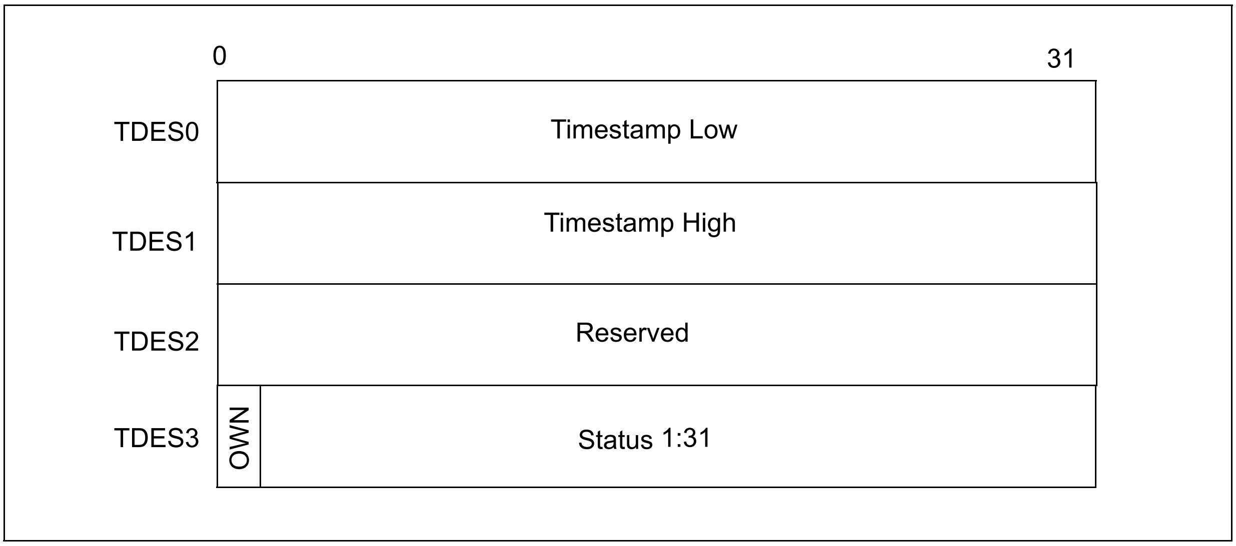
Table 910. TDES3 Normal Descriptor (Read Format) (continued)

Field	Description
14:15 CIC	Checksum Insertion Control These bits control the checksum calculation and insertion. The following list describes the bit encoding: 00 Checksum Insertion Disabled. 01 Only IP header checksum calculation and insertion are enabled. 10 IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 11 IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware.
16	Reserved
17:31 FL	Packet Length This field is equal to the length of the packet to be transmitted in bytes. When the TSE bit is not set, this field is equal to the total length of the packet to be transmitted: Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length

47.4.1.3.1.5 Transmit Normal Descriptor (Write-Back Format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[3]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 835. Transmit Descriptor Write-Back Format



47.4.1.3.1.6 TDES0 Normal Descriptor (Write-Back Format)

This format is only applicable to the last descriptor of a packet.

Table 911. TDES0 Normal Descriptor (Write-Back Format)

Field	Description
0:31 TTSL	Transmit Packet Timestamp Low The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

47.4.1.3.1.7 TDES1 Normal Descriptor (Write-Back Format)

This format is only applicable to the last descriptor of a packet.

Table 912. TDES1 Normal Descriptor (Write-Back Format)

Field	Description
0:31 TTSH	Transmit Packet Timestamp High The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

47.4.1.3.1.8 TDES2 Normal Descriptor (Write-Back Format)

This format is applicable only to the last descriptor of a packet.

Table 913. TDES2 Normal Descriptor (Write-Back Format)

Field	Description
0:31	Reserved

47.4.1.3.1.9 TDES3 Normal Descriptor (Write-Back Format)

This format is applicable only to the last descriptor of a packet.

Figure 836. TDES3 Normal Descriptor (Write-Back Format)

0	1	2	3	4:13	14	15	16	17	18	19	20	21	22	23	24:27	28	29	30	31
OWN	CTXT	FD	LD	Rsvd	TTSS	Rsvd	ES	JT	FF	PCE	LoC	NC	LC	EC	CC	ED	UF	DB	IHE

Table 914. TDES3 Normal Descriptor (Write-Back Format)

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 'b0.
1 CTXT	Context Type This bit should be set to 'b0 for Normal descriptor.
2 FD	First Descriptor This bit indicates that the buffer contains the first segment of a packet.
3 LD	Last Descriptor This bit is set 'b1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
4:13	Reserved
14 TTSS	Tx Timestamp Status This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES2 and TDES3 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [3]) in a descriptor is set. This bit is valid only when IEEE1588 timestamping feature is enabled; otherwise, it is reserved.
15	Reserved
16 ES	Error Summary This bit indicates the logical OR of the following bits: TDES3[31]: IP Header Error TDES3[17]: Jabber Timeout TDES3[18]: Packet Flush TDES3[19]: Payload Checksum Error TDES3[20]: Loss of Carrier TDES3[21]: No Carrier TDES3[22]: Late Collision TDES3[23]: Excessive Collision TDES3[28]: Excessive Deferral TDES3[29]: Underflow Error
17 JT	Jabber Timeout This bit indicates that the MAC transmitter has experienced a jabber time-out. This bit is set only when the JD bit of the MAC_Configuration register is not set.
18 FF	Packet Flushed This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.

Table 914. TDES3 Normal Descriptor (Write-Back Format) (continued)

Field	Description
19 PCE	Payload Checksum Error This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either because of insufficient bytes, as indicated by the Payload Length field of the IP Header or the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode.
20 LoC	Loss of Carrier This bit indicates that Loss of Carrier occurred during packet transmission (that is, the MII_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
21 NC	No Carrier This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
22 LC	Late Collision This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode). This bit is not valid if Underflow Error is set.
23 EC	Excessive Collision This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after first collision and the transmission of the packet is aborted.
24:27 CC	Collision Count This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
28 ED	Excessive Deferral This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000 Mbps mode or Jumbo Packet enabled mode) if DC bit is set in the MAC_Configuration register.
29 UF	Underflow Error This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty Transmit Buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the suspended state and sets the underflow bit corresponding to a queue in the MTL_Interrupt_Status register.

Table 914. TDES3 Normal Descriptor (Write-Back Format) (continued)

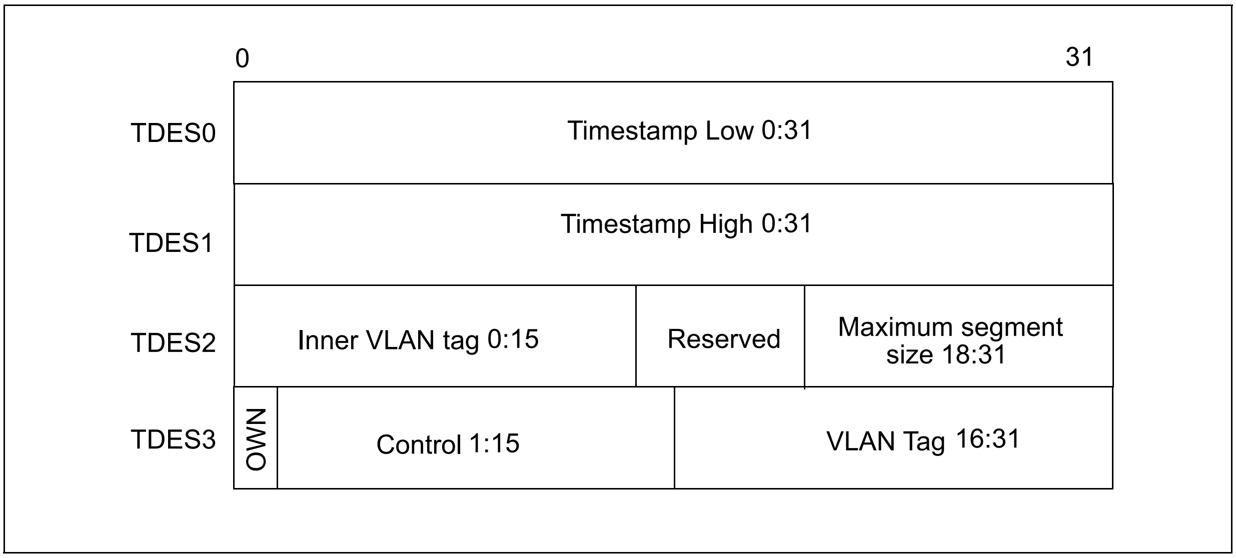
Field	Description
30 DB	Deferred Bit This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.
31 IHE	IP Header Error When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. This bit is valid only when Tx Checksum Offload is enabled. Otherwise, it is reserved. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

47.4.1.3.2 Transmit Context Descriptor

The Transmit Context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, VLAN Tag ID for VLAN insertion feature, and SA insertion bit for SA insertion. Write back is done on a context descriptor only to reset the OWN bit.

Note: The VLAN Tag IDs and MSS values, provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.
When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.
The Inner VLAN Tag Control input is used only for the next packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 837. Transmit Context Descriptor Format



47.4.1.3.2.1 TDES0 Context Descriptor

Table 915. TDES0 Context Descriptor

Field	Description
0:31 TTSL	Transmit Packet Timestamp Low For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

47.4.1.3.2.2 TDES1 Context Descriptor

Table 916. TDES1 Context Descriptor

Field	Description
0:31 TTSH	Transmit Packet Timestamp High For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

47.4.1.3.2.3 TDES2 Context Descriptor

Table 917. TDES2 Context Descriptor

Field	Description
0:15 IVT	Inner VLAN Tag When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[0:15] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
16:31	Reserved

47.4.1.3.2.4 TDES3 Context Descriptor

Figure 838. TDES3 Context Descriptor

0	1	2:3	4	5	6:7	8	9:13	14	15	16:31
OWN	CTXT	Rsvd	OSTC	TCMSSV	Rsvd	CDE	Rsvd	IVLTV	VLTV	VT

Table 918. TDES3 Context Descriptor

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
1 CTXT	Context Type This bit should be set to 1'b1 for Context descriptor.
2:3	Reserved
4 OSTC	One-Step Timestamp Correction Enable When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
5 TCMSSV	One-Step Timestamp Correction Input When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid.
6:7	Reserved
8 CDE	Context Descriptor Error When this bit is set, it indicates that the context descriptor was provided in the incorrect sequence and the DMA ignored it. The DMA sets this bit during write-back while closing the context descriptor.
9:11	Reserved
12:13 IVTIR	Inner VLAN Tag Insert or Replace When this bit is set, these bits request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. The following list describes the values of these bits: 00 Do not add the inner VLAN tag. 01 Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. 10 Insert an inner VLAN tag with the tag value programed in the MAC_VLAN_Incl register or context descriptor. 11 Replace the inner VLAN tag in packets with the tag value programed in the MAC_Inner_VLAN_Incl register or context descriptor. This option should be used only with the VLAN frames.
14 IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.
15 VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
16:13 VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the MAC_VLAN_Incl register is reset.

47.4.1.4 Receive Descriptor

The DMA in the Ethernet module attempts to read a descriptor only if the Tail Pointer is different from the Base Pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC. Otherwise, the performance of the DMA is impacted greatly because of the unavailability of the descriptors. In such situations, the RxFIFO in MTL becomes full and starts dropping packets.

The following Receive Descriptors are present:

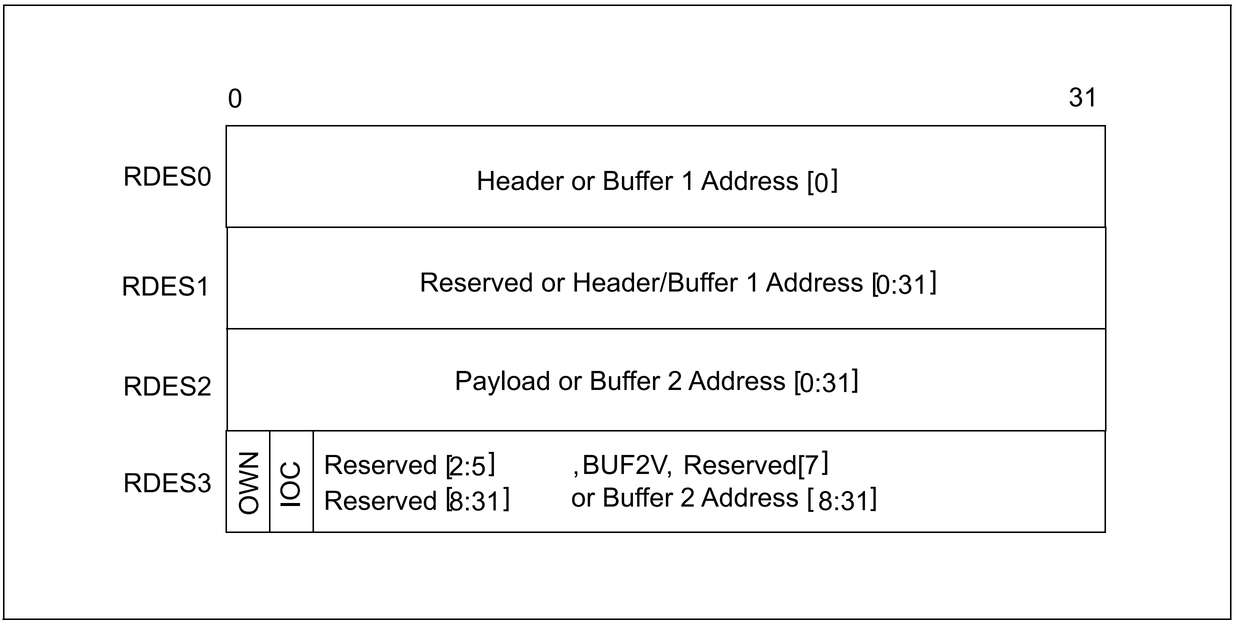
- Normal and
- Context descriptors

All RX descriptors are prepared by the software and given to the DMA as “Normal” Descriptors with the content as shown in *Section 47.4.1.4.1: Receive Normal Descriptor (Read Format)*. The DMA reads this descriptor and after transferring a received packet (or part of) to the buffers indicated by the descriptor, the Rx DMA will close the descriptor with the corresponding packet status. The format of this status is given in the *Section 47.4.1.4.2: Receive Normal Descriptor (Write-Back Format)*.

For some packets, the normal descriptor bits are not enough to write the complete status. For such packets, the RX DMA will write the extended status to the next descriptor (without processing or using the Buffers Pointers embedded in that descriptor). The format and content of this descriptor write back is described in *Section 47.4.1.4.3: Receive Context Descriptor*.

47.4.1.4.1 Receive Normal Descriptor (Read Format)

Figure 839. Receive Normal Descriptor (Read Format)



Note: In the Receive Descriptor (Read Format), if the Buffer Address field is all 0s, the Ethernet module does not transfer data to that buffer and skips to the next buffer or next descriptor.

47.4.1.4.1.1 RDES0 Normal Descriptor (Read Format)

Table 919. RDES0 Normal Descriptor (Read Format)

Field	Description
0:31 BUF1AP	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. The application can program a byte-aligned address for this buffer which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a Write operation with RDES0[29:31] as zero. However, the packet data is shifted as per actual offset as given by buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

47.4.1.4.1.2 RDES1 Normal Descriptor (Read Format)

Table 920. RDES1 Normal Descriptor (Read Format)

Field	Description
0:31	This field is reserved.

47.4.1.4.1.3 RDES2 Normal Descriptor (Read Format)

Table 921. RDES2 Normal Descriptor (Read Format)

Field	Description
0:31 BUF2AP	Buffer 2 Address Pointer These bits indicate the physical address of Buffer 2. RxDMA uses the LS Bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores BUF2AP[29:31] and writes to the complete location.

47.4.1.4.1.4 RDES3 Normal Descriptor (Read Format)

Figure 840. RDES3 Normal Descriptor (Read Format)

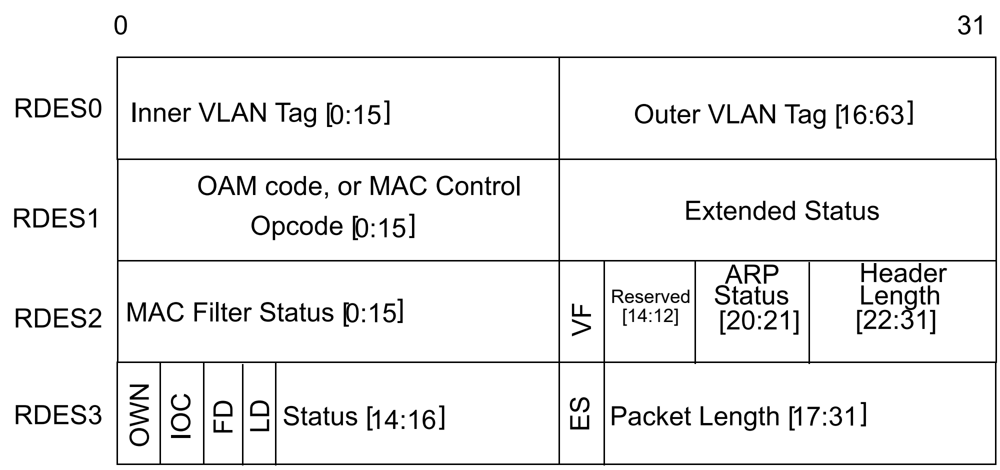
0	1	2:5	6	7	8:31
OWN	IOC	Rsvd	BUF2V	BUF1V	Rsvd

Table 922. RDES3 Normal Descriptor (Read Format)

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
1 IOC	Interrupt Enabled on Completion When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
2:5	Reserved
6 BUF2V	Buffer 2 Address Valid When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is Valid. The application must set this bit so that the DMA can use the address, to which the Buffer 2 address in RDES2 is pointing, to write received packet data.
7 BUF1V	Buffer 1 Address Valid When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by Buffer 1 address in RDES1 can be used by the DMA to write received packet data.
8:31	Reserved

47.4.1.4.2 Receive Normal Descriptor (Write-Back Format)

Figure 841. Receive Normal Descriptor (Write-Back Format)



47.4.1.4.2.1 RDES0 Normal Descriptor (Write-Back Format)

Figure 842. RDES0 Normal Descriptor (Write-Back Format)

0:15	16:31
IVT	OVT

Table 923. RDES0 Normal Descriptor (Write-Back Format)

Field	Description
0:15 IVT	Inner VLAN Tag This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
16:31 OVT	Outer VLAN Tag This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

47.4.1.4.2.2 RDES1 Normal Descriptor (Write-Back Format)

The status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

Figure 843. RDES1 Normal Descriptor (Write-Back Format)

0:15	16	17	18	19	20:23	24	25	26	27	28	29:31
OPC	TD	TSA	PV	PFT	PMT	IPCE	IPCB	IPV6	IPV4	IPHE	PT

Table 924. RDES1 Normal Descriptor (Write-Back Format)

Field	Description
0:15 OPC	OAM Sub-Type Code, or MAC Control Packet opcode OAM Sub-Type Code If bits[13:15] of RDES3 are set to 111, this field contains the OAM sub-type and code fields. MAC Control Packet opcode If bits[13:15] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.
16 TD	Timestamp Dropped This bit indicates that the timestamp was captured for this packet but it got dropped in the MTL Rx FIFO because of overflow. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.

Table 924. RDES1 Normal Descriptor (Write-Back Format) (continued)

Field	Description
17 TSA	Timestamp Available When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [3]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
18 PV	PTP Version This bit indicates that the received PTP message has the IEEE 1588 version 2 format. When this bit is reset, it indicates the IEEE 1588 version 1 format. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
19 PFT	PTP Packet Type This bit indicates that the PTP message is sent directly over Ethernet. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
20:23 PMT	PTP Message Type These bits are encoded to give the type of the message received: 0000 No PTP message received 0001 SYNC (all clock types) 0010 Follow_Up (all clock types) 0011 Delay_Req (all clock types) 0100 Delay_Resp (all clock types) 0101 Pdelay_Req (in peer-to-peer transparent clock) 0110 Pdelay_Resp (in peer-to-peer transparent clock) 0111 Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) 1000 Announce 1001 Management 1010 Signaling 1011–1110 Reserved 1111 PTP packet with Reserved message type These bits are available only when you select the Timestamp feature.
24 IPCE	IP Payload Error When this bit is set, it indicates either of the following: The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 16 (ES) of RDES3 is not set when this bit is set.
25 IPCB	IP Checksum Bypassed This bit indicates that the checksum offload engine is bypassed.
26 IPV6	IPv6 header Present This bit indicates that an IPV6 header is detected.
27 IPV4	IPV4 Header Present This bit indicates that an IPV4 header is detected.

Table 924. RDES1 Normal Descriptor (Write-Back Format) (continued)

Field	Description
28 IPHE	IP Header Error When this bit is set, it indicates either of the following: The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. The IP datagram version is not consistent with the Ethernet Type value. Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 26 or bit 27 is set.
29:31 PT	Payload Type These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): 000 Unknown type or IP/AV payload not processed 001 UDP 010 TCP 011 ICMP 110 AV Tagged Data Packet 111 AV Tagged Control Packet 101 AV Untagged Control Packet 100 IGMP if IPV4 Header Present bit is set else data center bridging (DCB) link layer discovery protocol (LLDP) Control Packet If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 000.

47.4.1.4.2.3 RDES2 Normal Descriptor (Write-Back Format)

Figure 844. RDES2 Normal Descriptor (Write-Back Format)

0:2	3	4	5:12	13	14	15	16	17:20	21	22:31
L3L4FM	L4FM	L3FM	MADRM	HF	DAF	SAF	VF	Rsvd	ARPNR	Rsvd

Table 925. RDES2 Normal Descriptor (Write-Back Format)

Field	Description
0:2 L3L4FM	Layer 3 and Layer 4 Filter Number Matched These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: 000 Filter 0 001 Filter 1 010 Filter 2 011 Filter 3 100 Filter 4 101 Filter 5 110 Filter 6 111 Filter 7 This field is valid only when Bit 3 or Bit 4 is set high. When more than one filter matches, these bits give the number of lowest filter.
3 L4FM	Layer 4 Filter Match When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: Layer 3 fields are not enabled and all enabled Layer 4 fields match All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by Bits[0:2].
4 L3FM	Layer 3 Filter Match When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by Bits[0:2].
5:12 MADRM	MAC Address Match or Hash Value When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the hash value computed by the MAC. A packet passes the hash filter when the bit corresponding to the hash value is set in the hash filter register.
13 HF	Hash Filter Status When this bit is set, it indicates that the packet passed the MAC address hash filter. Bits[5:12] indicate the hash value.
14 DAF	Destination Address Filter Fail When this bit is set, it indicates that the packet failed the DA Filter in the MAC.
15 SAF	SA Address Filter Fail When this bit is set, it indicates that the packet failed the SA Filter in the MAC.
16 VF	VLAN Filter Status When this bit is set, it indicates that the VLAN Tag of received packet passed the VLAN filter.

Table 925. RDES2 Normal Descriptor (Write-Back Format) (continued)

Field	Description
17:20	Reserved
21 ARPNR	ARP Reply Not Generated When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).
22:31	Reserved

47.4.1.4.2.4 RDES3 Normal Descriptor (Write-Back Format)

Figure 845. RDES3 Normal Descriptor (Write-Back Format)

0	1	2	3	4	5	6	7	8	9	10	11	12	13:15	16	17:31
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT	ES	PL

Table 926. RDES3 Normal Descriptor (Write-Back Format)

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
1 CTXT	Receive Context Descriptor When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 1'b0 to this bit for normal receive descriptor.
2 FD	First Descriptor When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet.
3 LD	Last Descriptor When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
4 RS2V	Receive Status RDES2 Valid When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
5 RS1V	Receive Status RDES1 Valid When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
6 RS0V	Receive Status RDES0 Valid When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.

Table 926. RDES3 Normal Descriptor (Write-Back Format) (continued)

Field	Description
7 CE	CRC Error When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
8 GP	Giant Packet When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). **Note:** Giant packet indicates only the packet length. It does not cause any packet truncation.
9 RWT	Receive Watchdog Timeout When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
10 OE	Overflow Error When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. **Note:** This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
11 RE	Receive Error When this bit is set, it indicates that the MII_RXER signal is asserted while the MII_RX_DV signal is asserted during packet reception. This error also includes carrier extension error in the MII and half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
12 DE	Dribble Bit Error When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
13:15 LT	Length/Type Field This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: 000 The packet is a length packet 001 The packet is a type packet. 011 The packet is a ARP Request packet type 100 The packet is a type packet with VLAN Tag 101 The packet is a type packet with Double VLAN Tag 110 The packet is a MAC Control packet type 111 The packet is a OAM packet type 010 Reserved

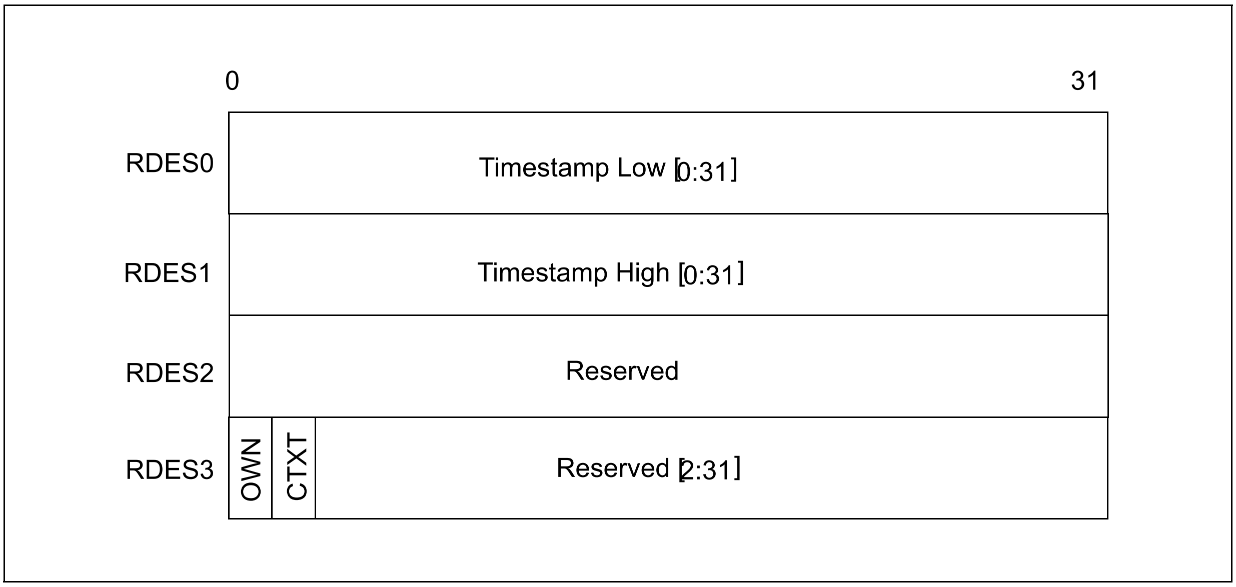
Table 926. RDES3 Normal Descriptor (Write-Back Format) (continued)

Field	Description
16 ES	Error Summary When this bit is set, it indicates the logical OR of the following bits: RDES3[7]: CRC Error RDES3[12]: Dribble Error RDES3[11]: Receive Error RDES3[9]: Watchdog Timeout RDES3[10]: Overflow Error RDES3[8]: Giant Packet This field is valid only when the LD bit of RDES3 is set.
17:31 PL	Packet Length These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and either the Descriptor Error (RDES3[18]) or Overflow Error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current packet.

47.4.1.4.3 Receive Context Descriptor

This descriptor is read-only for the application. Only DMA can write to this descriptor. The context descriptor provides information about the extended status related to the last received packet. The Bit 1 of RDES3 indicates the context type descriptor.

Figure 846. Receive Context Descriptor



47.4.1.4.3.1 RDES0 Context Descriptor

Table 927. RDES0 Context Descriptor

Field	Description
0:31 RTSL	Receive Packet Timestamp Low The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

47.4.1.4.3.2 RDES1 Context Descriptor

Table 928. RDES1 Context Descriptor

Field	Description
0:31 RTSH	Receive Packet Timestamp High The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

47.4.1.4.3.3 RDES2 Context Descriptor

Table 929. RDES2 Context Descriptor

Field	Description
0:31	Reserved

47.4.1.4.3.4 RDES3 Context Descriptor

Figure 847. RDES3 Context Descriptor

0	1	2:31
OWN	CTXT	Rsvd

Table 930. RDES3 Context Descriptor

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
1 CTXT	Receive Context Descriptor When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor.
2:31	Reserved

47.5 Programming Guidelines

This section provides the instructions for initializing the DMA or MAC registers in the proper sequence.

47.5.1 Initializing DMA

Complete the following steps to initialize the DMA:

1. Provide a software reset. This resets all of the MAC internal registers and logic (bit 31 of *DMA_Mode Register (DMA_MODE)*)
2. Wait for the completion of the reset process (poll bit 31 of the *DMA_Mode Register (DMA_MODE)*, which is only cleared after the reset operation is completed)
3. Program the following fields to initialize the *System Bus Mode Register (DMA_SYSBUS_MODE)*:
   a) AAL
   b) Fixed burst or undefined burst
   c) Burst mode values for AHB bus interface
4. Create a descriptor list for transmit and receive. In addition, ensure that the receive descriptors are owned by DMA (set bit 0 of descriptor TDES3/RDES3)

Note: Descriptor address from start to end of the ring should not cross the 4 GB boundary.

5. Program the Transmit and Receive Ring length registers_ (DMA_CH*n*_TxDesc_Ring_Length and DMA_CH*n*_RxDesc_Ring_Length). The ring length programed must be at least 4
6. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (DMA_CH*n*_TxDesc_List_Address, DMA_CH*n*_RxDesc_List_Address). Also, program transmit and receive tail pointer

registers indicating to the DMA about the available descriptors (DMA_CH*n*_TxDesc_Tail_Pointer and DMA_CH*n*_RxDesc_Tail_Pointer)

7. Program the settings of the following registers for the parameters such as maximum burst-length (PBL) initiated by DMA, descriptor skip lengths, OSP in case of TxDMA, RBSZ in case of RxDMA, and so on:
   a) DMA_CH*n*_TX_Control
   b) DMA_CH*n*_TX_Control
   c) DMA_CH*n*_TX_Control
8. Enable the interrupts by programming the DMA_CH*n*_Interrupt_Enable register
9. Start the Receive and Transmit DMAs by setting SR (bit 31) of the DMA_CH*n*_RX_Control and ST (bit 31) of the DMA_CH*n*_TX_Control register
10. Repeat steps *4* to *9* for all the Tx DMA and Rx DMA channels selected in the hardware

47.5.2 Initializing MTL Registers

Complete the following steps to initialize the MTL Registers:

1. Program the Tx Scheduling (SCHALG) and Receive Arbitration Algorithm (RAA) fields in MTL_Operation_Mode to initialize the MTL operation for multiple Tx and Rx queues
2. Program the Receive Queue to DMA mapping in the MTL_RxQ_DMA_Map0 register
3. Program the following fields to initialize the mode of operation in the MTL_TxQ0_Operation_Mode register:
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) in case of threshold mode
   b) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue0
   c) Transmit Queue Size (TQS)
4. Program the following fields to initialize the mode of operation in the MTL_RxQ0_Operation_Mode register:
   a) Receive Store and Forward (RSF) or RTC in case of Threshold mode
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD)
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP)
   d) Receive Queue Size (RQS)
5. Repeat previous two steps for the second MTL Tx and Rx queue

47.5.3 Initializing MAC

The following MAC Initialization operations can be performed after DMA initialization. If the MAC initialization is completed before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: MAC_Address0_High and MAC_Address0_Low. For more than one MAC address program the MAC addresses appropriately
2. Program the following fields to set the appropriate filters for the incoming frames in the MAC_Packet_Filter register:
   a) Receive All
   b) Promiscuous mode
   c) Hash or Perfect Filter
   d) Unicast, multicast, broadcast, and control frames filter settings
3. Program the following fields for proper flow control in the MAC_Q0_Tx_Flow_Ctrl register:
   a) Pause time and other Pause frame control bits
   b) Transmit Flow control bits
   c) Flow Control Busy
4. Program the MAC_Interrupt_Enable register, as required
5. Program the appropriate fields in the MAC_Configuration register. For ex: Inter-packet gap while transmission and jabber disable.
6. Set bit 31 and 30 in MAC_Configuration registers to start the MAC transmitter and receiver

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the Host (either transmit or receive)
2. Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and writing the descriptor tail pointer to Tx/Rx tail pointer register
   a) DMA_CH*n*_TxDesc_Tail_Pointer
   b) DMA_CH*n*_RxDesc_Tail_Pointer
4. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process
   a) DMA_CH*n*_Current_App_TxDesc
   b) DMA_CH*n*_Current_App_RxDesc
5. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process
   a) DMA_CH*n*_Current_App_TxBuffer
   b) DMA_CH*n*_Current_App_RxBuffer

47.5.3.1 For Stopping and Starting Transmission

Complete the following steps to pause the transmission for some time.

Note: The steps are provided for Channel 0.

1. Disable the transmit DMA (if applicable) by clearing Bit 31 (ST) of the DMA_CH*n*_TX_Control register
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of the MTL_TxQ0_Debug register (TRCSTS is not 01 and TXQSTS=0)
3. Disable the MAC transmitter and MAC receiver by clearing Bit 31 (RE) and Bit 30(TE) of the MAC_Configuration register
4. Disable the receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of the MTL_RxQ0_Debug register, PRXQ=0 and RXQSTS=00)
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in the MTL_TxQ0_Debug register and RXQSTS is 0 in the MTL_RxQ0_Debug register)
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver

Note: Do not change the configuration (such as duplex mode, speed, port, or loop back) when the MAC is actively transmitting or receiving. The Software should change these parameters only when the MAC transmitter and receiver are not active.
Similarly the DMA related configuration should not be changed when Transmit and Receive DMA are in active.

47.5.4 Programming Guidelines for Multi-Channel Multi-Queuing

Following are the steps for Transmit:

1. Program the Transmit queue size in the TQS field of MTL_TxQ*n*_Operation_Mode register. Based on the value programed in TQS field, the size of the queue is determined. In the Transmit operation, the number of channels is equal to the number of the queues. Due to this reason, the Channel to Queue mapping is fixed
2. Enable ST bit of DMA_CH*n*_TX_Control register and corresponding TXQEN in MTL_TxQ*n*_Operation_Mode register
3. The scheduling method need to be programed in SCHALG of the MTL_Operation_Mode register
4. In case of CBS algorithm in AVB queues, the following registers also needs to be programed as required:
   a) MTL_TxQ*n*_ETS_Control
   b) MTL_TxQ*n*_SendSlopeCredit
   c) MTL_TxQ*n*_HiCredit
   d) MTL_TxQ*n*_LoCredit

Following are the steps for Receive:

1. Program the Receive queue size in the RQS field of MTL_RxQ*n*_Operation_Mode Register. Based on the value programed in RQS field, the size of the queue is determined
2. Enable the Receive Queues 0 to 1 in the fields RXQ0EN to RXQ1EN in MAC_RxQ_Ctrl0 Register for AV. In DMA configurations, SR bit of statically or

dynamically mapped DMA_CH*n*_RX_Control Register and corresponding RXQ*n*_EN in MAC_RxQ_Ctrl0 Register needs to be enabled

3. The MAC routes the Rx packets to the Rx Queues based on following packet types:
   a) AV PTP Packets: Based on the programming of AVPTPQ in MAC_RxQ_Ctrl1 Register
   b) AV Untagged Control packets: Based on the programming of AVCPQ in MAC_RxQ_Ctrl1 register
   c) VLAN Tag Priority field in VLAN Tagged packets: Program PSRQ1-0 of the MAC_RxQ_Ctrl2 register for the routing of tagged packets based on the USP (user Priority) field of the received packets to the Rx Queues 0 to 1
   d) The AV tagged control and data packets are also routed based on PSRQ field of the MAC_RxQ_Ctrl2 register
4. If multiple RX DMA channels are enabled, the following programming should be done for proper arbitration and mapping:
   a) Program the RAA field of MTL_Operation_Mode register to select the arbitration algorithm to decide which RxQ is read out from the RxFIFO memory
   b) Program the MTL_RxQ*n*_Control to decide the weights and the packet arbitration for each RxQ
   c) If static mapping is programed in MTL_RxQ_DMA_Map0 register (RXQ*n*DADMACH is reset to 0), bits RXQx2DMA and others need to be programed to select the channel for which each queue is mapped
   d) Set RXQ*n*DADMACH bit in MTL_RxQ_DMA_Map0 Register to select dynamic mapping of packets in each RxQueue
   e) In dynamic channel mapping, the routing of a packet to a specific RxDMA channel is decided by the value of DCS field in the lowest MAC Address Register

Note: The priorities set in PSRQ1-0 should be unique.

47.5.5 Programming Guidelines for MII Link State Transitions

The following steps are to be performed when the link is down but the Transmit and Receive clocks are running:

Note: The steps are provided for Channel 0.

1. Disable the Transmit DMA (if applicable) by clearing Bit 31 (ST) of DMA_CH*n*_TX_Control Register
2. Disable the MAC receiver by clearing Bit 29 (RE) of the MAC_Configuration register
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of MTL_TxQ0_Debug register (TRCSTS is not 01).
   -or-
   Flush the Tx FIFO for faster empty operation
4. Disable the MAC transmitter by clearing Bit 30 (TE) of the MAC_Configuration register
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in the MTL_TxQ0_Debug register and RXQSTS is 0 in the MTL_RxQ0_Debug register)
6. After the link is up, read the PHY registers to know the latest configuration and accordingly program the MAC registers
7. Restart the operation by starting the Tx DMA, and then enabling the MAC Transmitter and Receiver

You do not need to disable the Rx DMA. As the Receiver is disabled, the FIFO does not get any data in the Rx FIFO.

Do the following steps when the link is down and the Transmit and Receive clocks are stopped:

Note: The steps are provided for Channel 0.

1. Disable the MAC Transmitter and Receiver by clearing bits RE & TE of MAC_Configuration register. This will not take effect immediately as the clocks are absent
2. Wait till the link is up and the clocks are restored
3. Wait for the completion of the transfer of any partial frame if any at time of stopping of Transmit/Receive clock. This can be checked by reading the MAC_Debug Register (should be all-zero). Some old packets may still remain in the TXFIFO as the MAC Transmitter is stopped
4. Read the PHY registers to know the latest operating mode and accordingly program the MAC registers
5. Restart the MAC Transmitter and Receiver by setting RE & TE bits

47.5.6 Programming Guidelines for IEEE 1588 Timestamping

47.5.6.1 Initialization Guideline for System Time Generation

The timestamp feature is enabled by setting Bit 31 of the MAC_Timestamp_Control register. However, it is essential that the timestamp counter should be initialized after this bit is set. Complete the following steps during for initialization:

1. Mask the Timestamp Trigger interrupt by clearing the bit 15 of MAC_Interrupt_Enable Register
2. Set Bit 31 of MAC_Timestamp_Control register to enable timestamping
3. Program MAC_Sub_Second_Increment Register based on the PTP clock frequency
4. If you are using the Fine Correction approach, program MAC_Timestamp_Addend and set Bit 26 of MAC_Timestamp_Control register
5. Poll the MAC_Timestamp_Control register until Bit 26 is cleared
6. Program Bit 30 of MAC_Timestamp_Control register to select the Fine Update method (if required)
7. Program MAC_System_Time_Seconds_Update Register and MAC_System_Time_Nanoseconds_Update Register with the appropriate time value
8. Set Bit 29 in MAC_Timestamp_Control register.
   The timestamp counter starts operation as soon as it is initialized with the value written in the Timestamp Update registers.
   If one-step timestamping is enabled:
   a) To enable one-step timestamping, program Bit 4 of the TDES Context Descriptor
   b) Program registers MAC_Timestamp_Ingress_Asym_Corr and MAC_Timestamp_Egress_Asym_Corr to update the correction field in PDelay_Req PTP messages
9. Enable the MAC receiver and transmitter for proper timestamping

Note: If timestamp operation is disabled by clearing Bit 31 of MAC_Timestamp_Control Register, you need to repeat all these steps to restart the timestamp operation.

47.5.6.2 System Time Correction

To synchronize or update the system time in one process (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the Timestamp Update registers (MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update)
2. Set Bit 28 (TSUPDT) of the MAC_Timestamp_Control Register.
   The value in the Timestamp Update registers is added to or subtracted from the system time when the TSUPDT bit is cleared

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm explained in System Time Register Module, calculate the rate by which you want to make the system time increments slower or faster.
2. Update the MAC_Timestamp_Addend with the new value and set Bit 26 of the MAC_Timestamp_Control Register
3. Wait for the time for which you want the new value of the Addend register to be active. You can do this by enabling the Timestamp Trigger interrupt after the system time reaches the target value
4. Program the required target time in the MAC_PPS0_Target_Time_Seconds register and the MAC_PPS0_Target_Time_Nanoseconds register
5. Enable the Timestamp interrupt in bit 19 of MAC_Interrupt_Enable register
6. Set bit 27 in Register MAC_Timestamp_Control
7. When this trigger causes an interrupt, read MAC_Interrupt_Status Register
8. Reprogram MAC_Timestamp_Addend Register with the old value and set bit 26 again

47.5.7 Programming Guidelines for AV Feature

Complete the following steps to initialize the DMA for AV Feature:

1. Provide a software reset to reset all QOS internal registers and logic (bit 31 in DMA_Mode register)
2. Wait for the completion of the reset process. Poll bit 31 of the DMA_Mode register, which is cleared only after the reset operation is completed
3. Program the fields to initialize the DMA register by setting the values in DMA_Mode register
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the Transmit and Receive descriptors. When OSF mode is used, at least two TX descriptors are required
5. Make sure that your software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors
6. Program the Transmit and Receive Ring length registers. The ring length programed must be at least 4. Then initialize receive and transmit descriptor list address with the

base address of transmit and receive descriptor. Finally program the Transmit and Receive tail pointer registers indicating to the DMA about the available descriptors:

a) DMA_CH*n*_TxDesc_Ring_Length

b) DMA_CH*n*_RxDesc_Ring_Length

c) DMA_CH*n*_TxDesc_List_Address

d) DMA_CH*n*_RxDesc_List_Address

e) DMA_CH*n*_TxDesc_Tail_Pointer

f) DMA_CHn_RxDesc_Tail_Pointer

7. Program the following fields to initialize the mode of operation in the MTL_TxQ1_Operation_Mode register:
   a) Transmit Store And Forward (TSF)
   b) Transmit Threshold Control (TTC)
   c) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue0
   d) Transmit Queue Size (TQS)
8. Enable the interrupts by programming the DMA_CH*n*_Interrupt_Enable register
9. Repeat steps *4* through *9* for all additional channels of AV feature.
10. Program the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV Queues
11. Start the Receive and Transmit DMA by setting bit 31 of the DMA_CH*n*_TX_Control register and bit 31 of the DMA_CH*n*_RX_Control register

47.5.7.1 Enabling Slot Number Checking

The slot number check feature is used to specify the intervals at which the DMA Channels mapped to AV Queues fetches the frames from the AHB system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the host memory. The feature is available only when you enable time-stamping and program the MAC_Sub_Second_Increment register. Complete the following steps to enable the slot number checking:

Note: These steps should be completed after Step 10 and before Step 11 of Section 47.5.7: Programming Guidelines for AV Feature.

1. Enable time-stamping by following the steps described in *Section 47.5.6.1: Initialization Guideline for System Time Generation*
2. Make sure that the SLOTNUM field (bits 9:12) of TDES3 Normal Descriptor (Read Format) contains a valid slot number. You can read the current reference slot number from the DMA_CH*n*_Slot_Function_Control_Status
3. Set Bit 31 (ESC) of the Slot Function Control and Status register of a channel to enable the slot number checking

47.5.7.2 Enabling Average Bits Per Slot Reporting

The CBS Status register of the additional AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting:

1. Enable time-stamping by following the steps described in *Section 47.5.6.1: Initialization Guideline for System Time Generation*
2. Program Bits [25:27], SLC, of the MTL_TxQ*n*_ETS_Control register of a channel with number of slots over which the average transmitted bits per slot need to be computed
3. Enable Bit 22 (ABPSSIE) of the MTL_Q*n*_Interrupt_Control_Status register of a channel to generate the average bits per slot interrupt
4. Read Bits [15:31], ABS, from the MTL_TxQ*n*_ETS_Status register of a channel on each interrupt

Note: The software can read the ABS bits in polling mode even if the ABPSIE bit is not enabled. When high, bit 30 (ABPSIS) of the MTL_TxQn_ETS_Status register indicates that a new value is updated in the ABS field.

The frequency of this interrupt depends on the value programed in the Step 2. For example, when you program value 0 in the SLC field, the interrupt is generated at every 125 microsecond.
When not required, you can disable this interrupt to stop the interrupt flooding.

47.5.8 Programming Guidelines for Energy Efficient Ethernet

47.5.8.1 Entering and Exiting the Tx LPI Mode

For entering and exiting the Tx LPI Mode, complete the following steps:

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode)
3. Program Bits[6:15] and Bits[16:31] in MAC_LPI_Timers_Control register
4. Read the link status of the PHY chip by using the MDIO interface and update Bit 14 of MAC_LPI_Control_Status accordingly. This update should be done whenever the link status in the PHY chip changes
5. Program the MAC_1US_Tic_Counter as per the frequency of the clock used for accessing the CSR slave port
6. Program the MAC_LPI_Entry_Timer register (LPIET) with the IDLE time for which the MAC should wait before entering the LPI state on its own
7. Set LPIET and LPITXA (bit[11:12]) of MAC_LPI_Control_Status register to enable the auto-entry into LPI and auto-exit of MAC from LPI state
8. Set Bit 15 of MAC_LPI_Control_Status register to make the MAC Transmitter enter the LPI state.
   The MAC enters the LPI mode after completing all scheduled packets and remains IDLE for the time indicated by LPIET. It sets the TLPIEN (bit[31]) after entry to LPI state
9. When a packet is scheduled for transmission (when the TxDMA comes out of IDLE state or when a packet is presented at ATI or MTI interface), the MAC Transmitter exits LPI state automatically. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission
10. MAC Transmitter re-enters LPI state if it remains IDLE for LPIET time and sets the TLPIEN bit and the entry-exit cycle continues
11. Reset LPITXEN in case the application wants to over-ride the auto-entry/exit modes and make the MAC Transmitter exit the LPI state directly

Note: To make the MAC enter the LPI state only after it completes the transmission of all queued frames in the Tx FIFO, you should set Bit 12 in MAC_LPI_Control_Status register.
To switch off the MII Transmit Clock during the LPI state, use the sbd_tx_clk_gating_ctrl_o signal for gating the clock input.
To switch off the CSR clock or power to the rest of the system during the LPI state, you should wait for the TLPIEN interrupt of MAC_LPI_Control_Status register to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

47.5.8.2 Gating Off the CSR Clock in the LPI Mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

47.5.8.3 Gating Off the CSR Clock in the Rx LPI Mode

The following operations are performed when the MAC receives the LPI pattern from the PHY.

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status [RLPIEN interrupt of MAC_LPI_Control_Status register is set
2. The interrupt pin is asserted. The interrupt is cleared when the host reads the MAC_LPI_Control_Status register

After the interrupt is asserted and the MAC Tx is also in the LPI mode, you can gate-off the CSR clock. If the MAC TX is not in the LPI mode when you gate off the CSR clock, the events on the MAC transmitter do not get reported or updated in the CSR.

For restoring the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when MAC_LPI_Control_Status register is read.

47.5.8.4 Gating Off the CSR Clock in the Tx LPI Mode

The following operations are performed when Bit 15 (LPIEN) of MAC_LPI_Control_Status register is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of MAC_LPI_Control_Status register) is set
2. The interrupt pin is asserted. The interrupt is cleared when the host reads the MAC_LPI_Control_Status register

After the interrupt is asserted and the MAC RX is also in the LPI mode, you can gate off the CSR clock. If the MAC RX is not in the LPI mode when you gate off the CSR clock, the events on the MAC receiver do not get reported or updated in the CSR.

For restoring the CSR clock, switch on the CSR clock when the MAC has to come out of the TX LPI mode. After the CSR clock is resumed, reset Bit 15 (LPIEN) of MAC_LPI_Control_Status register to bring the MAC out of the LPI mode.

47.5.9 Programming Guidelines for Flexible Pulse-Per-Second (PPS) Output

47.5.9.1 Generating single Pulse on PPS

1. Program 11 or 10 (for interrupt) in Bits [25:26], TRGTMODSEL, of the MAC_PPS_Control register. This instructs the MAC to use the Target Time registers

(MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) for start time of PPS signal output

2. Program the start time value in the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds)
3. Program the width of the PPS signal output in MAC_PPS0_Width Register
4. Program Bits [28:31], PPSCMD, of MAC_PPS_Control to 0001. This instructs the MAC to generate single pulse on the PPS signal output at the time programed in the Target Time registers

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programed start time elapses. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be more than the time at which the falling edge occurs for the previous pulse
2. Program the width of the next PPS signal output in MAC_PPS0_Width
3. Program Bits [28:3], PPSCMD, of MAC_PPS_Control to generate a single pulse after the time at which the previous pulse is de-asserted. This instructs the MAC to generate single pulse on the PPS signal output, at the time programed in Target Time registers

If you give this command before the previous pulse becomes low, then the new command overwrites the previous command and the QOS may generate only 1 extended pulse.

47.5.9.2 Generating a pulse train on PPS

1. Program 11 or 10 (for interrupt) in Bits [25:26], TRGTMODSEL, of MAC_PPS_Control register. This instructs the MAC to use the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) for start time of the PPS signal output
2. Program the start time value in the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds)
3. Program the interval value between the train of pulses on the PPS signal output in MAC_PPS0_Interval Register
4. Program the width of the PPS signal output in MAC_PPS0_Width Register
5. Program Bits[28:3], PPSCMD, of MAC_PPS_Control Register to 0010. This instructs the MAC to generate train of pulses on the PPS signal output with start time programed in Target Time registers.
   By default, the PPS pulse train is free-running unless stopped by ‘STOP Pulse train at time’ or ‘STOP Pulse Train immediately’ commands.
6. Program the stop value in the Target Time registers. Ensure that Bit 0 (TSTRBUSY) of MAC_PPS0_Target_Time_Nanoseconds Register is reset before programming the Target Time registers again.
7. Program the PPSCMD field (bit 28:31) of MAC_PPS_Control to 0100. This stops the train of pulses on PPS signal output after the programed stop time specified in Step *6* elapses.

You can stop the pulse train at any time by programming 0101 in the PPSCMD field. Similarly, you can cancel the Stop Pulse train command (given in Step 7) by programming 0110 in the PPSCMD field before the time (programed in Step 6) elapses. You can cancel the pulse train generation by programming 0011 in the PPSCMD field before the programed start time (in Step 2) elapses.

47.5.9.3 Generating an Interrupt without Affecting the PPS

The Bits [25:26], TRGTMODSEL, of the MAC_PPS_Control Register enable you to program the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) to do any one of the following:

- Generate only interrupts
- Generate interrupts and the PPS start and stop time
- Generate only PPS start and stop time

To program the Target Time registers to generate only interrupt event:

1. Program 00 (for interrupt) in Bits [25:26], TRGTMODSEL, of MAC_PPS_Control register. This instructs the MAC to use the Target Time registers for target time interrupt
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.
   If Bits [25:26], TRGTMODSEL, are changed (for example, to control the PPS), then the interrupt generation is over-written with the new mode and new programed Target Time register value

47.5.10 Programming Guidelines for VLAN filtering on Receive

1. Program MAC_VLAN_Tag Register for the following bit to select the filtering method:
   a) ETV: Enable 12-Bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   VTHM: VLAN Tag Hash Table Match Enable
   b) ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   c) ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   d) DOVLTC: Ignores VLAN Type for Tag Match
   e) VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL of MAC_VLAN_Tag register for the 12 bit or 16 bit VLAN tag.
3. If Hash filtering of VLAN tag is enabled, program MAC_VLAN_Hash_Table Register. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table

48 Inter-Integrated Circuit (I2C)

48.1 Overview

This chapter describes the Inter-Integrated Circuit (I^2C) bus module implemented on this chip and presents the following topics:

- *Section 48.2: Introduction to I2C*
- *Section 48.3: External signal descriptions*
- *Section 48.4: Memory map and register definition*
- *Section 48.5: Functional description*
- *Section 48.6: Initialization and application information*

48.2 Introduction to I2C

This section presents the following topics:

- *Section 48.2.1: Definition: I2C module*
- *Section 48.2.2: Advantages of the I2C bus*
- *Section 48.2.3: Module block diagram*
- *Section 48.2.4: Features*
- *Section 48.2.5: Modes of operation*
- *Section 48.2.6: Definition: I2C conditions*

48.2.1 Definition: I2C module

The I^2C module is a functional unit that provides a two-wire – serial data (SDA) and serial clock (SCL) – bidirectional serial bus that provides a simple and efficient method of data exchange between this chip and other devices, such as microcontrollers, EEPROMs, real-time clock devices, analog-to-digital converters, and LCDs.

48.2.2 Advantages of the I2C bus

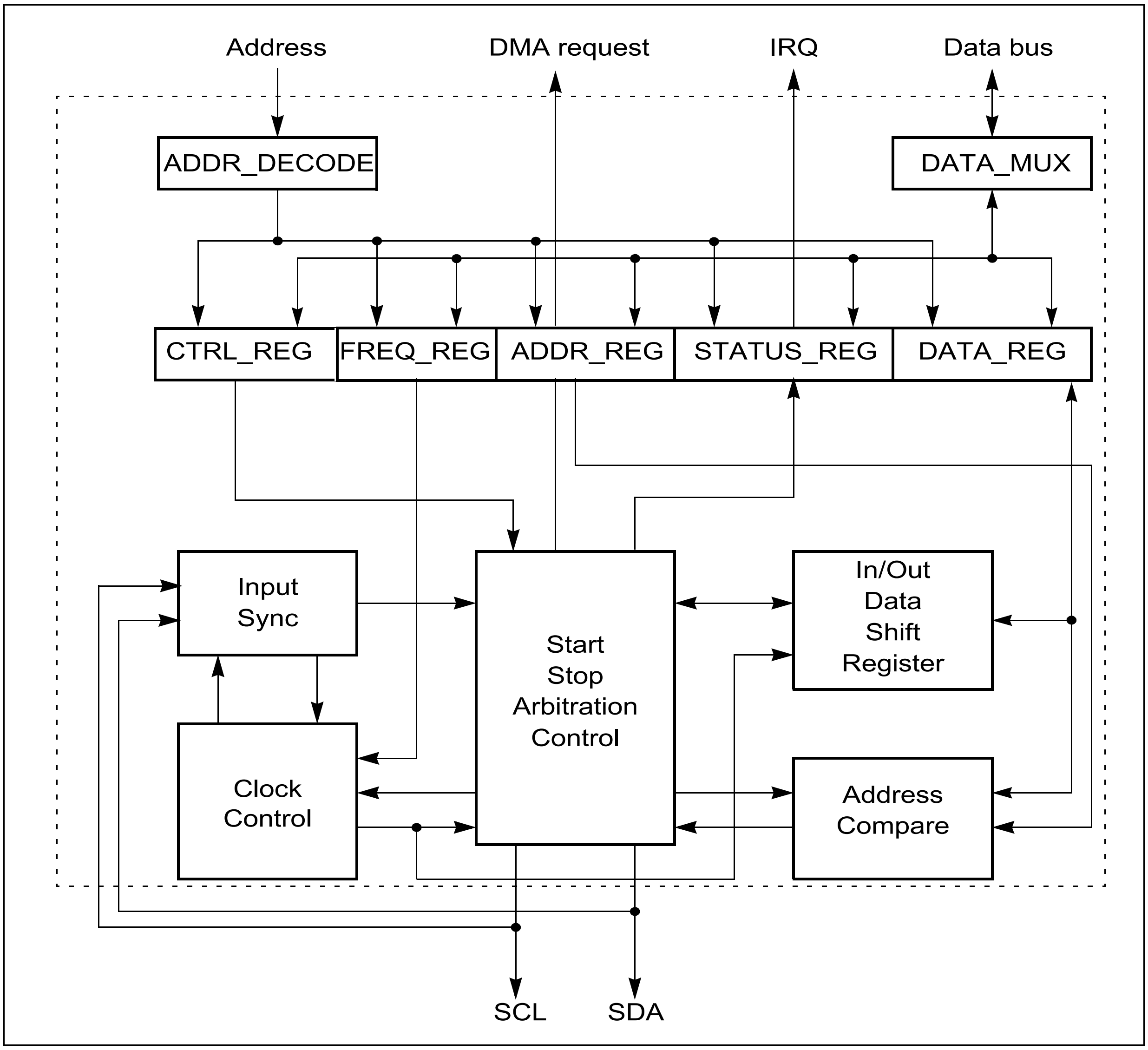
The synchronous, multiple-master two-wire I^2C bus:

- Minimizes interconnections between devices
- Allows the connection of additional devices to the bus for expansion and system development
- Includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously
- Does not require an external address decoder

48.2.3 Module block diagram

Figure 848 shows a block diagram of the I^2C module.

Figure 848. I^{2}C block diagram



48.2.4 Features

The I^{2}C module has the following key features:

- Compatible with I^{2}C bus standard[r]
- Operating speeds
  – Up to 100 kbps in Standard Mode
  – Up to 400 kbps in Fast Mode
  – Actual baud rate dependent on the SCL rise time (which depends on external pull-up resistor values and bus loading)
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Basic DMA interface
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF

48.2.5 Modes of operation

The I^{2}C module supports the chip modes described in *Table 931*.

Table 931. Chip modes supported by the I^{2}C module

Chip mode	Description	Important notes
RUN	Basic mode of operation	—
STOP	The lowest-power mode that allows the chip to turn off all the clocks to the I^{2}C module	The I^{2}C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus. Refer to *Section 48.5.6: IPG STOP mode*.
IPG DEBUG	Allows the chip to freeze all ongoing activities (such as an ongoing transaction, counter values, and register status) for debugging.	Refer to *Section 48.5.7: IPG DEBUG mode*.

In addition to chip modes, the I^{2}C module has several module-specific modes. These are described in *Table 932*.

r. Compliant with I^{2}C 2.0 standard with the exception that HS (high speed) mode is not supported

Table 932. Module-specific modes supported by the I^{2}C module

Module mode	Description	Important notes
Master mode	The I^{2}C module is the driver of the SDA line.	– Do not use the I^{2}C module's slave address as a calling address. – The I^{2}C module cannot be a master and a slave simultaneously.
Slave mode	The I^{2}C module is not the driver of the SDA line.	– Enable the I^{2}C module before a START condition from a non-I^{2}C master is detected. – By default the I^{2}C module performs as a slave receiver.

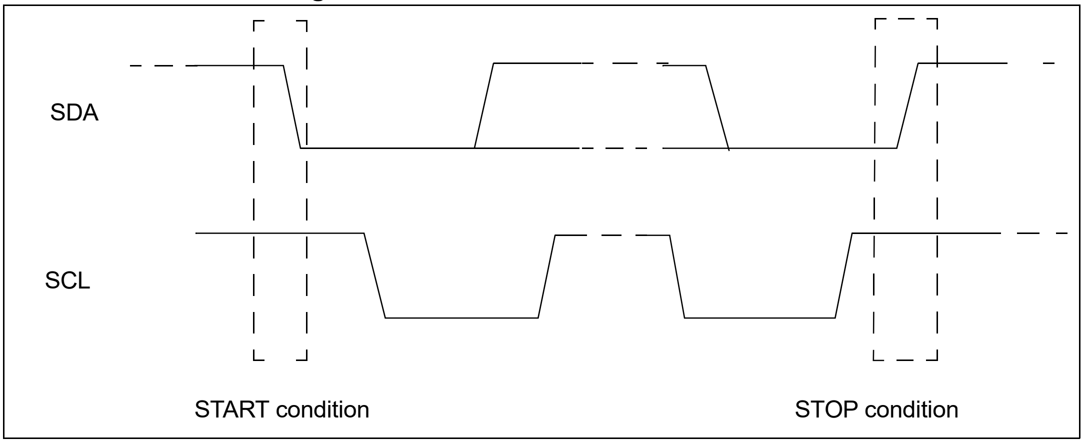
48.2.6 Definition: I^{2}C conditions

Table 933 shows the I^{2}C-specific conditions defined for the I^{2}C module.

Table 933. I^{2}C conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data. It is defined as a high-to-low transition of SDA while SCL is high, as shown in *Figure 849*.
STOP	A condition generated by the master to terminate a transfer and free the bus. It is defined as a low-to-high transition of SDA while SCL is high, as shown in *Figure 849*.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.

Figure 849. START and STOP conditions



48.3 External signal descriptions

This section presents the following topics:

- Section 48.3.1: Signal overview
- Section 48.3.2: Detailed external signal descriptions

48.3.1 Signal overview

The I^2C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The logical AND function is performed on both signals with external pull-up resistors. For the electrical characteristics of these signals, refer to the data sheet for this chip.

48.3.2 Detailed external signal descriptions

The SDA and SCL signals are described in *Table 934*.

Table 934. External signal descriptions

Signal	Description
SCL	Bidirectional serial clock line of the module, compatible with the I^2C bus specification
SDA	Bidirectional serial data line of the module, compatible with the I^2C bus specification

48.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the I^2C module. It presents the following topics:

- Section 48.4.1: I2C memory map
- Section 48.4.2.1: Register accessibility
- Section 48.4.2.2: I2C Bus Address register (IBAD)
- Section 48.4.2.3: I2C Bus Frequency Divider register (IBFD)
- Section 48.4.2.4: I2C Bus Control Register (IBCR)
- Section 48.4.2.5: I2C Bus Status Register (IBSR)
- Section 48.4.2.6: I2C Bus Data I/O Register (IBDR)
- Section 48.4.2.7: I2C Bus Interrupt Config register (IBIC)
- Section 48.4.2.8: I2C Bus Debug register (IBDBG)

48.4.1 I^2C memory map

The memory map for the I^2C module is given below in *Table 935*. The total address for each register is the sum of the base address for the I^2C module and the address offset for each register.

Table 935. I²C memory map

Address offset	Register	Section
0x0000	I²C Bus Address Register (IBAD)	Section 48.4.2.2
0x0001	I²C Bus Frequency Divider Register (IBFD)	Section 48.4.2.3
0x0002	I²C Bus Control Register (IBCR)	Section 48.4.2.4
0x0003	I²C Bus Status Register (IBSR)	Section 48.4.2.5
0x0004	I²C Bus Data I/O Register (IBDR)	Section 48.4.2.6
0x0005	I²C Bus Interrupt Config Register (IBIC)	Section 48.4.2.7
0x0006	I²C Bus Debug Register (IBDBG)	Section 48.4.2.8
0x0007–0x3FFF	Reserved(1)	

1. If enabled at the SoC level, reads or writes to these registers will cause bus aborts. Refer to the System Services Module documentation for more details.

48.4.2 Registers definition

48.4.2.1 Register accessibility

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries.

As an example, the IBFD is at address offset 0x01, and can be accessed in any of the following ways:

- 8-bit R/W access of address offset 0x0001
- Second byte of 16-bit R/W access of address offset 0x0000
- Second byte of 32-bit R/W access of address offset 0x0000

The IBFD register cannot be accessed by a 16- or 32-bit RW operation at address offset 0x0001 because those operations require an address aligned to a 16- or 32-bit boundary.

48.4.2.2 I²C Bus Address register (IBAD)

This register contains the address the I²C Bus will respond to when addressed as a slave. This is not the address sent on the bus during the address transfer.

Offset: 0x0000

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

	0	1	2	3	4	5	6	7
R	ADR							0
W								
Reset	0	0	0	0	0	0	0	0

Figure 850. I²C Bus Address register (IBAD)

Table 936. IBAD field descriptions

Field	Description
0:6 ADR	Slave Address Specific slave address to be used by the I^{2}C Bus module. **Note:** The default mode of I^{2}C Bus is slave mode for an address match on the bus.

48.4.2.3 I^{2}C Bus Frequency Divider register (IBFD)

Offset: 0x0001

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

	0	1	2	3	4	5	6	7
R W	IBC							
Reset	0	0	0	0	0	0	0	0

Figure 851. I^{2}C Bus Frequency Divider register (IBFD)

Table 937. IBFD field descriptions

Field	Description
0:7 IBC	I-Bus Clock Rate This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider. The IBC bits are decoded to give the Tap and Prescale values as follows: 0:1 Selects the prescaled shift register (refer to *Table 946*) 2:4 Selects the prescaler divider (refer to *Table 947*) 5:7 Selects the shift register tap point (refer to *Table 948*)

48.4.2.4 I^{2}C Bus Control Register (IBCR)

Offset: 0x0002

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

	0	1	2	3	4	5	6	7
R	MDIS	IBIE	MSSL	TXRX	NOACK	0	DMAEN	0
W						RSTA		
Reset	1	0	0	0	0	0	0	0

Figure 852. I^{2}C Bus Control Register (IBCR)

Table 938. IBCR field descriptions

Field	Description
0 MDIS	Module disable This bit controls the software reset of the entire I^{2}C Bus module. 0 The I^{2}C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect. 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed. Status register bits (IBSR) are not valid when module is disabled. **Note:** If the I^{2}C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I^{2}C Bus module losing arbitration, after which, bus operation would return to normal.
1 IBIE	I-Bus Interrupt Enable 0 Interrupts from the I^{2}C Bus module are disabled. This does not clear any currently pending interrupt condition. 1 Interrupts from the I^{2}C Bus module are enabled. An I^{2}C Bus interrupt occurs provided the IBIF bit in the status register is also set.
2 MSSL	Master/Slave mode select When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. This field is cleared without generating a STOP signal when the master loses arbitration. 0 Slave mode. 1 Master mode.
3 TXRX	Transmit/Receive mode select This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive. 1 Transmit.
4 NOACK	Data Acknowledge disable This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I^{2}C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. **Note:** Values written to this bit are only used when the I^{2}C Bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, acknowledge bit = 1).

Table 938. IBCR field descriptions (continued)

Field	Description
5 RSTA	Repeat Start Writing a one to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 0 No effect. 1 Generate repeat start cycle.
6 DMAEN	DMA Enable When this bit is set, the DMA Tx and Rx lines will be asserted when the I^2C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the I^2C module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. Refer to the DMA Application Information section for more details. 0 Disable the DMA TX/RX request signals. 1 Enable the DMA TX/RX request signals.

48.4.2.5 I^2C Bus Status Register (IBSR)

Offset: 0x0003

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

	0	1	2	3	4	5	6	7
R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
W				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

Figure 853. I^2C Bus Status Register (IBSR)

Table 939. IBSR field descriptions

Field	Description
0 TCF	Transfer complete While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. **Note:** This bit is only valid during or immediately following a transfer to the I^2C module or from the I^2C module. 0 Transfer in progress 1 Transfer complete
1 IAAS	Addressed as a slave When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set the TXRX field accordingly. Writing to the I-Bus Control Register clears this bit. 0 Not addressed 1 Addressed as a slave
2 IBB	Bus busy This bit indicates the status of the bus. When a START signal is detected, IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. 0 Bus is Idle 1 Bus is busy
3 IBAL	Arbitration Lost The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: – SDA is sampled low when the master drives a high during an address or data transmit cycle. – SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. – A start cycle is attempted when the bus is busy. – A repeated start cycle is requested in slave mode. – A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
5 SRW	Slave Read/Write When the IAAS bit is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By reading this field, the CPU can detect slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave

Table 939. IBSR field descriptions (continued)

Field	Description
6 IBIF	I-Bus Interrupt Flag The IBIF bit is set when one of the following conditions occurs: – Arbitration lost (IBAL bit set) – Byte transfer complete (TCF bit set and DMAEN bit not set) – Addressed as slave (IAAS bit set) – NoAck from Slave (MS and Tx bits set) – I^2C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. In DMA mode (DMAEN set) a byte transfer complete condition will not trigger the setting of IBIF. All other conditions still apply.
7 RXAK	Received Acknowledge This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete. 0 Acknowledge received 1 No acknowledge received

48.4.2.6 I^2C Bus Data I/O Register (IBDR)

In master transmit mode, when data is written to the IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.

Note: The IBCR[TXRX] field must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the most recent byte received while the I^2C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I^2C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MSSL is used for the address transfer and should comprise the calling address (in position DATA[7:1]) concatenated with the required $R/\overline{W}$ bit (in position D0).

Note: When the I^2C is configured in master mode and receiving data from a slave that is transmitting data bytes on an irregular basis, the master cannot know whether the data received in the IBDR is the old latched data or the new data received from the slave. To avoid this, 2 consecutive intermittent data bytes from slave should be different.

Offset: 0x0004

Access:
Supervisor mode only
(user mode accesses are ignored, and
no error response is asserted while
accessing this register in user mode)

	0	1	2	3	4	5	6	7
R W	DATA							
Reset	0	0	0	0	0	0	0	0

Figure 854. I^2C Bus Data I/O Register (IBDR)

Table 940. IBDR field descriptions

Field	Description
0:7 DATA	Data transmitted or received

48.4.2.7 I^2C Bus Interrupt Config register (IBIC)

Offset: 0x0005

Access:
Supervisor mode only
(user mode accesses are ignored, and
no error response is asserted while
accessing this register in user mode)

	0	1	2	3	4	5	6	7
R	BIIE	BYTERXIE	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 855. I^2C Bus Interrupt Config register (IBIC)

Table 941. IBIC field descriptions

Field	Description
0 BIIE	Bus Idle Interrupt Enable bit This config bit can be used to enable the generation of an interrupt once the I^2C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I^2C bus. 0 Bus Idle Interrupts disabled 1 Bus Idle Interrupts enabled
1 BYTERXIE	Byte receive interrupt enable This field is used to generate an interrupt every time the I^2C master/slave receives a new byte. This feature can be useful when an I^2C master is receiving data from a slave that is transmitting on an irregular basis.

48.4.2.8 I^{2}C Bus Debug register (IBDBG)

Offset: 0x0006

Access:
Supervisor mode only
(user mode accesses are ignored, and
no error response is asserted while
accessing this register in user mode)

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	IPG_DEBUG_HALTED	IPG_DEBUG_EN
W								
Reset	0	0	0	0	0	0	0	0

Figure 856. I^{2}C Bus Debug register (IBDBG)

Table 942. IBDBG field descriptions

Field	Description
6 IPG_DEBUG_HALTED	Debug Halted Bit This is s status bit which can be read back after asserting the debug enable signal so as to know if the IP has entered the debug mode or not. 0 IP is still executing a transaction 1 IP has entered the debug mode
7 IPG_DEBUG_EN	Debug enable bit This bit is used to enable IP enter the debug mode provided the IPG DEBUG signal is high. All the registers, counter values and status bits are frozen and can be accessed by the CPU. 0 Normal operation, Bus Idle Interrupts disabled 1 IP is in debug mode **Note:** If the assertion of this bit along with the IPG DEBUG signal happens in the middle of a transaction, IP enters the debug mode only after successful completion of the current transaction after which no further transaction can take place until the debug mode is exited.

48.5 Functional description

This section presents the following topics:

- *Section 48.5.1: Notes about module operation*
- *Section 48.5.2: Transactions*
- *Section 48.5.3: Arbitration procedure*
- *Section 48.5.4: Clock behavior*
- *Section 48.5.5: Interrupts*
- *Section 48.5.7: IPG DEBUG mode*
- *Section 48.5.8: DMA interface*

48.5.1 Notes about module operation

- The I^2C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I^2C module is acting as a master, it must not try to call its own slave address.

48.5.2 Transactions

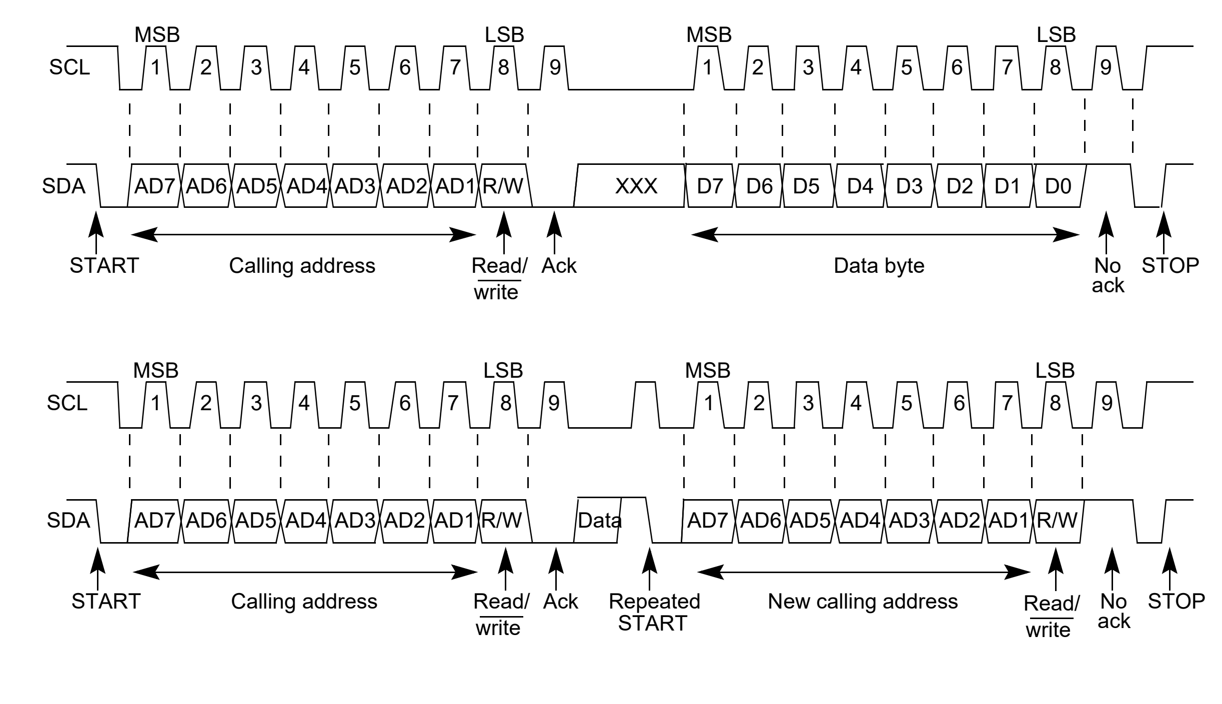
This section presents the following topics:

- *Section 48.5.2.1: Protocol overview*
- *Section 48.5.2.2: Transaction protocol definitions*
- *Section 48.5.2.4: High-level protocol steps*
- *Section 48.5.2.5: START condition*
- *Section 48.5.2.6: Slave address transmission*
- *Section 48.5.2.7: Data transmission*
- *Section 48.5.2.8: STOP condition*
- *Section 48.5.2.9: Repeated START condition*

48.5.2.1 Protocol overview

Figure 857 shows the behavior of SCL and SDA during a typical I^2C transaction.

Figure 857. I²C transaction protocol



48.5.2.2 Transaction protocol definitions

This section defines several important terms presented in *Figure 857*.

Table 943. I²C definitions

Term	Definition
START	A START condition, as defined in *Section 48.2.6: Definition: I2C conditions*
STOP	A STOP condition, as defined in *Section 48.2.6: Definition: I2C conditions*
Calling (slave) address	A seven-bit address used to identify a slave on the I²C bus. The requirements for specifying this address are presented in *Section 48.5.2.3: I2C calling address requirements*.
Read/write (R/$\overline{W}$)	A bit that specifies the direction of the data transfer to the slave as follows: 0 The data is being transferred from the master to the slave (“write”). 1 The data is being transferred from the slave to the master (“read”).
Ack	A bit that specifies the acknowledgment of a calling address, indicated by pulling SDA low.

48.5.2.3 I²C calling address requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

48.5.2.4 High-level protocol steps

The I^2C protocol conceptually supports two types of transfers, which are illustrated in *Figure 857*. The significant steps in these transfers are presented in *Table 944*. Details of each of these steps are presented in subsequent sections.

Table 944. I^2C high-level protocol steps

Standard transfer	Repeated START transfer
1. START condition 2. Slave target or general call address transmission 3. Acknowledgment from slave 4. Data transfer 5. STOP condition 6. (repeat Steps 1–4)	1. START condition 2. Slave target or general call address transmission 3. Acknowledgment from slave 4. Data transfer 5. Repeated START condition 6. (repeat Steps 2–4 as needed) 7. STOP condition 8. (repeat Steps 1–7)

48.5.2.5 START condition

When the bus is free, that is no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START condition (refer to *Section 48.2.6*). This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

48.5.2.6 Slave address transmission

The master transmits the slave address on the next clock cycle after the START condition (refer to *Section 48.5.2.5*). The process of slave address transmission is presented in *Table 945*.

Table 945. Slave address transmission process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the $R/\overline{W}$ bit.
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: – The acknowledge bit is set: The master must initiate a data transfer followed by either a STOP condition or a repeated START condition. – The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

48.5.2.7 Data transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data
- Awakens all slaves
- Proceeds on a byte-by-byte basis in the direction specified by the $R/\overline{W}$ bit sent by the calling master

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.
- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.
- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

48.5.2.8 STOP condition

The master can terminate the communication by generating a STOP condition (refer to *Section 48.2.6*). It can do so even if the slave has generated an acknowledge, at which point the slave must release the bus.

A master is not required to send a STOP condition at the end of every transfer. For more information, refer to *Section 48.5.2.9*.

48.5.2.9 Repeated START condition

The I^2C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of *Figure 857*.

48.5.3 Arbitration procedure

The I^2C bus is a true multi-master bus that allows more than one master to be connected to it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic “1” while another master transmits logic “0”. The losing masters immediately switch over to slave receive mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

48.5.4 Clock behavior

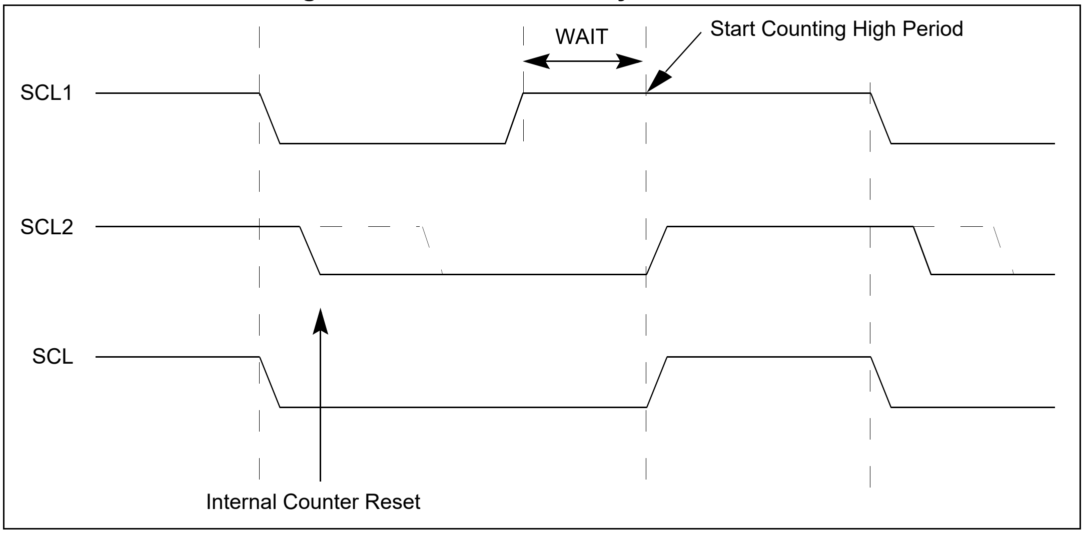
This section presents the following topics:

- *Section 48.5.4.1: Clock synchronization*
- *Section 48.5.4.2: Clock stretching*
- *Section 48.5.4.3: Handshaking*
- *Section 48.5.4.4: Clock rate and IBFD settings*

48.5.4.1 Clock synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (refer to *Figure 858*). When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

Figure 858. I²C bus clock synchronization



48.5.4.2 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

48.5.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

48.5.4.4 Clock rate and IBFD settings

The following tables describe the settings of several fields in the IBFD register (refer to *Section 48.4.2.3: I2C Bus Frequency Divider register (IBFD)*).

Table 946. I-Bus multiplier factor

IBC[0:1]	MUL
00	01
01	02
10	04
11	Reserved

Table 947. I-Bus prescaler divider values

IBC[2:4]	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

Table 948. I-Bus tap and prescale values

IBC[5:7]	SCL tap (clocks)	SDA tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of *Table 947*. All subsequent tap points are separated by 2^{IBC} as shown in the tap2tap column in *Table 947*. The SCL Tap is used to generate the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to the change of state of SDA that is the SDA Hold time.

Figure 859. SDA Hold time

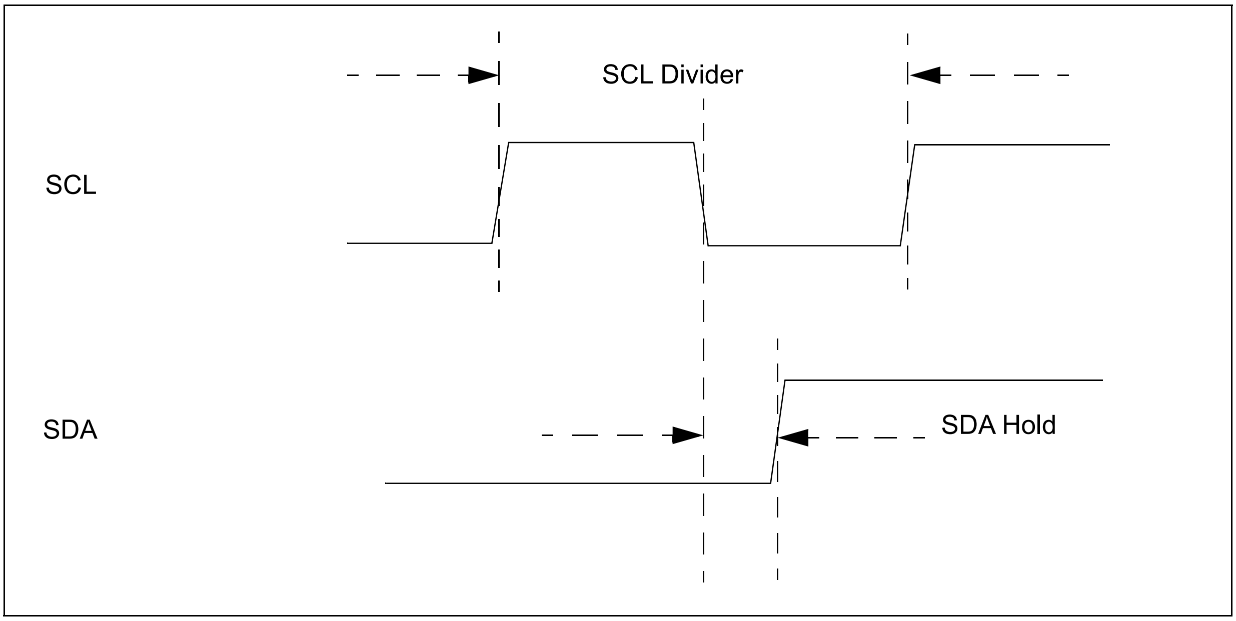
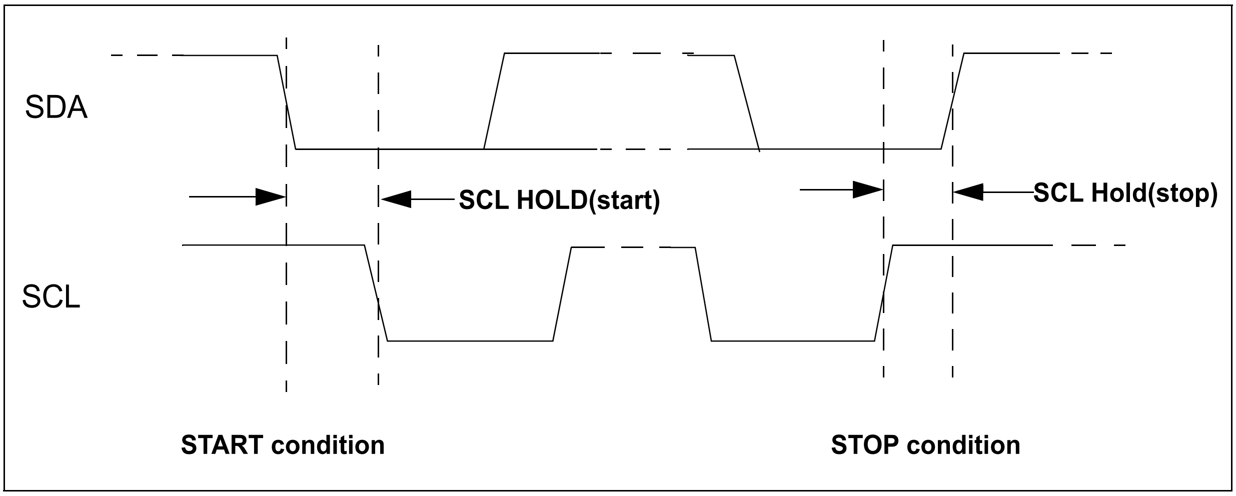


Figure 860. SCL divider and SDA Hold



The equation used to generate the divider values from the IBFD bits is:

Equation 32 SCL Divider = MUL x {2 x (scl2tap + [(SCL_Tap -1) x tap2tap] + 2)}

The SDA Hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in *Table 949*. The equation used to generate the SDA Hold value from the IBFD bits is:

Equation 33 SDA Hold = MUL x {scl2tap + [(SDA_Tap - 1) x tap2tap] + 3}

The equations for SCL Hold values to generate the START and STOP conditions from the IBFD bits are:

Equation 34 SCL Hold(start) = MUL x [scl2start + (SCL_Tap - 1) x tap2tap]

Equation 35 SCL Hold(stop) = MUL x [scl2stop + (SCL_Tap - 1) x tap2tap]

Table 949. I^{2}C divider and hold values

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
1	0x00	20	7	6	11
1	0x01	22	7	7	12
1	0x02	24	8	8	13
1	0x03	26	8	9	14
1	0x04	28	9	10	15
1	0x05	30	9	11	16
1	0x06	34	10	13	18
1	0x07	40	10	16	21
1	0x08	28	7	10	15
1	0x09	32	7	12	17
1	0x0A	36	9	14	19
1	0x0B	40	9	16	21
1	0x0C	44	11	18	23
1	0x0D	48	11	20	25
1	0x0E	56	13	24	29
1	0x0F	68	13	30	35
1	0x10	48	9	18	25
1	0x11	56	9	22	29
1	0x12	64	13	26	33
1	0x13	72	13	30	37
1	0x14	80	17	34	41
1	0x15	88	17	38	45
1	0x16	104	21	46	53
1	0x17	128	21	58	65
1	0x18	80	9	38	41
1	0x19	96	9	46	49
1	0x1A	112	17	54	57
1	0x1B	128	17	62	65
1	0x1C	144	25	70	73
1	0x1D	160	25	78	81
1	0x1E	192	33	94	97
1	0x1F	240	33	118	121
1	0x20	160	17	78	81

Table 949. I^2C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
1	0x21	192	17	94	97
1	0x22	224	33	110	113
1	0x23	256	33	126	129
1	0x24	288	49	142	145
1	0x25	320	49	158	161
1	0x26	384	65	190	193
1	0x27	480	65	238	241
1	0x28	320	33	158	161
1	0x29	384	33	190	193
1	0x2A	448	65	222	225
1	0x2B	512	65	254	257
1	0x2C	576	97	286	289
1	0x2D	640	97	318	321
1	0x2E	768	129	382	385
1	0x2F	960	129	478	481
1	0x30	640	65	318	321
1	0x31	768	65	382	385
1	0x32	896	129	446	449
1	0x33	1024	129	510	513
1	0x34	1152	193	574	577
1	0x35	1280	193	638	641
1	0x36	1536	257	766	769
1	0x37	1920	257	958	961
1	0x38	1280	129	638	641
1	0x39	1536	129	766	769
1	0x3A	1792	257	894	897
1	0x3B	2048	257	1022	1025
1	0x3C	2304	385	1150	1153
1	0x3D	2560	385	1278	1281
1	0x3E	3072	513	1534	1537
1	0x3F	3840	513	1918	1921
2	0x40	40	14	12	22
2	0x41	44	14	14	24
2	0x42	48	16	16	26

Table 949. I^2C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
2	0x43	52	16	18	28
2	0x44	56	18	20	30
2	0x45	60	18	22	32
2	0x46	68	20	26	36
2	0x47	80	20	32	42
2	0x48	56	14	20	30
2	0x49	64	14	24	34
2	0x4A	72	18	28	38
2	0x4B	80	18	32	42
2	0x4C	88	22	36	46
2	0x4D	96	22	40	50
2	0x4E	112	26	48	58
2	0x4F	136	26	60	70
2	0x50	96	18	36	50
2	0x51	112	18	44	58
2	0x52	128	26	52	66
2	0x53	144	26	60	74
2	0x54	160	34	68	82
2	0x55	176	34	76	90
2	0x56	208	42	92	106
2	0x57	256	42	116	130
2	0x58	160	18	76	82
2	0x59	192	18	92	98
2	0x5A	224	34	108	114
2	0x5B	256	34	124	130
2	0x5C	288	50	140	146
2	0x5D	320	50	156	162
2	0x5E	384	66	188	194
2	0x5F	480	66	236	242
2	0x60	320	34	156	162
2	0x61	384	34	188	194
2	0x62	448	66	220	226
2	0x63	512	66	252	258
2	0x64	576	98	284	290

Table 949. I²C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
2	0x65	640	98	316	322
2	0x66	768	130	380	386
2	0x67	960	130	476	482
2	0x68	640	66	316	322
2	0x69	768	66	380	386
2	0x6A	896	130	444	450
2	0x6B	1024	130	508	514
2	0x6C	1152	194	572	578
2	0x6D	1280	194	636	642
2	0x6E	1536	258	764	770
2	0x6F	1920	258	956	96
2	0x70	1280	130	636	642
2	0x71	1536	130	764	770
2	0x72	1792	258	892	898
2	0x73	2048	258	1020	1026
2	0x74	2304	386	1148	1154
2	0x75	2560	386	1276	1282
2	0x76	3072	514	1532	1538
2	0x77	3840	514	1916	1922
2	0x78	2560	258	1276	1282
2	0x79	3072	258	1532	1538
2	0x7A	3584	514	1788	1794
2	0x7B	4096	514	2044	2050
2	0x7C	4608	770	2300	2306
2	0x7D	5120	770	2556	2562
2	0x7E	6144	1026	3068	3074
2	0x7F	7680	1026	3836	3842
4	0x80	80	28	24	44
4	0x81	88	28	28	48
4	0x82	96	32	32	52
4	0x83	104	32	36	56
4	0x84	112	36	40	60
4	0x85	120	36	44	64
4	0x86	136	40	52	72

Table 949. I^2C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
4	0x87	160	40	64	84
4	0x88	112	28	40	60
4	0x89	128	28	48	68
4	0x8A	144	36	56	76
4	0x8B	160	36	64	84
4	0x8C	176	44	72	92
4	0x8D	192	44	80	100
4	0x8E	224	52	96	116
4	0x8F	272	52	120	140
4	0x90	192	36	72	100
4	0x91	224	36	88	116
4	0x92	256	52	104	132
4	0x93	288	52	120	148
4	0x94	320	68	136	164
4	0x95	352	68	152	180
4	0x96	416	84	184	212
4	0x97	512	84	232	260
4	0x98	320	36	152	164
4	0x99	384	36	184	196
4	0x9A	448	68	216	228
4	0x9B	512	68	248	260
4	0x9C	576	100	280	292
4	0x9D	640	100	312	324
4	0x9E	768	132	376	388
4	0x9F	960	132	472	484
4	0xA0	640	68	312	324
4	0xA1	768	68	376	388
4	0xA2	896	132	440	452
4	0xA3	1024	132	504	516
4	0xA4	1152	196	568	580
4	0xA5	1280	196	632	644
4	0xA6	1536	260	760	772
4	0xA7	1920	260	952	964
4	0xA8	1280	132	632	644

Table 949. I^{2}C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
4	0xA9	1536	132	760	772
4	0xAA	1792	260	888	900
4	0xAB	2048	260	1016	1028
4	0xAC	2304	388	1144	1156
4	0xAD	2560	388	1272	1284
4	0xAE	3072	516	1528	1540
4	0xAF	3840	516	1912	1924
4	0x30	2560	260	1272	1284
4	0xB1	3072	260	1528	1540
4	0xB2	3584	516	1784	1796
4	0xB3	4096	516	2040	2052
4	0xB4	4608	772	2296	2308
4	0xB5	5120	772	2552	2564
4	0xB6	6144	1028	3064	3076
4	0xB7	7680	1028	3832	3844
4	0xB8	5120	516	2552	2564
4	0xB9	6144	516	3064	3076
4	0xBA	7168	1028	3576	3588
4	0xBB	8192	1028	4088	4100
4	0xBC	9216	1540	4600	4612
4	0xBD	10240	1540	5112	5124
4	0xBE	12288	2052	6136	6148
4	0xBF	15360	2052	7672	7684

48.5.5 Interrupts

This section presents the following topics:

- *Section 48.5.5.1: Interrupt vector*
- *Section 48.5.5.2: Interrupt description*

48.5.5.1 Interrupt vector

The I^{2}C module uses only one interrupt vector.

Table 950. Interrupt summary

Interrupt	Offset	Vector	Priority	Source	Description
I^{2}C Interrupt	—	—	—	IBAL, TCF, IAAS, IBB bits in IBSR register	When any of IBAL, TCF or IAAS bits is set, an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the de-assertion of IBB can also cause an interrupt, indicating that the bus is idle.

48.5.5.2 Interrupt description

There are five types of internal interrupts in the I^{2}C. The interrupt service routine can determine the interrupt type by reading the Status register.

I^{2}C Interrupt can be generated on the following events:

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set and DMAEN bit not set)
- Address Detect condition (IAAS bit set)
- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I^{2}C interrupt is enabled by the IBCR[IBIE] bit. It must be cleared by writing ‘1’ to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the IBIC[BIIE] bit.

48.5.6 IPG STOP mode

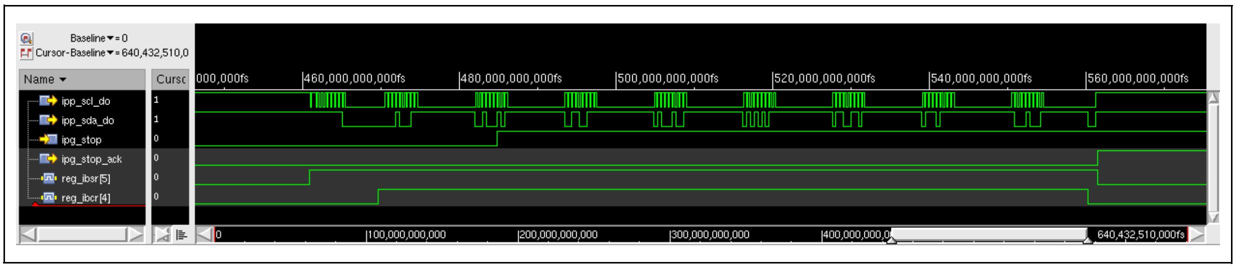
This mode allows the software to put the I^{2}C module in power-down state. Once the STOP request is asserted, the I^{2}C module comes to a graceful halt after completing all the ongoing transactions.

As soon as the I^{2}C module enters IPG STOP mode:

- The I^{2}C clock is disabled.
- No transaction can take place.
- All registers are inaccessible.

The I^{2}C module enters this mode only after successfully completing the current ongoing transaction, hence the low-power request is acknowledged once the STOP condition occurs. The user must ensure that the bus is free when STOP is requested. To request STOP for ongoing transmission the user must wait until the transmission is complete, followed by clearing of the IBCR[TXRX] field. Refer to *Figure 861* for more details.

Figure 861. I^{2}C stop mode behavior when master is in Rx and slave is in Tx



48.5.7 IPG DEBUG mode

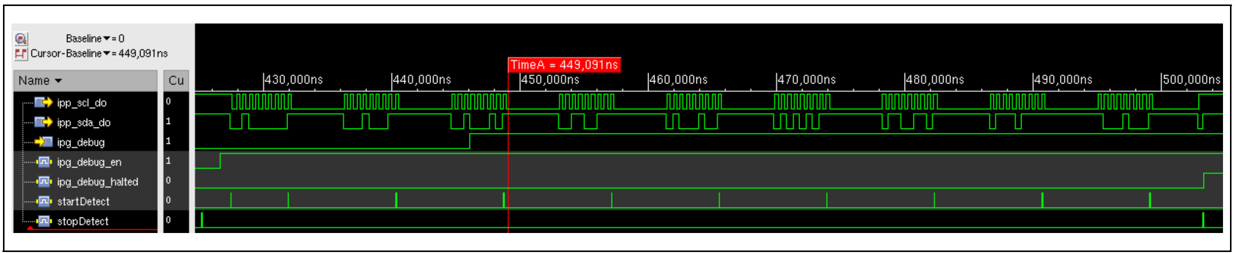
This mode allows CPU to debug the I^{2}C by freezing all the counters, registers and status bits. Once the Debug request is asserted along with IBDBG[IPG_DEBUG_EN] bit, I^{2}C comes to a graceful halt after completing all the ongoing transactions. A Debug halted signal IBDBG[IPG_DEBUG_HALTED] is also asserted to indicate that the debug request has been successfully serviced and is de-asserted when the Debug request is de-asserted.

As soon as the I^{2}C module enters IPG DEBUG mode:

- No transaction can take place.
- All the registers, counters and status bits are frozen. They all can be accessed by the CPU to enable the debugging.

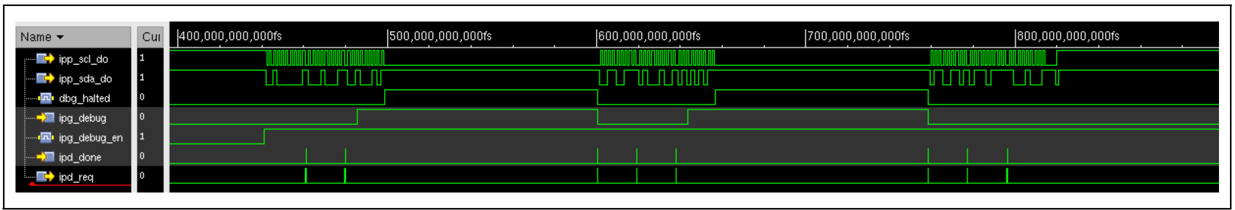
The I^{2}C module enters this mode only after successfully completing the current ongoing transaction. For example, if the current transaction consists of 8 bytes and the debug mode was initiated by user at the time of the second byte, the IPG Debug Halted signal will be asserted after the 8th byte is transmitted/received. Refer to the simulation result in *Figure 862* for more details.

Figure 862. Simulation result of IPG Debug Halted in case of frame transmission



If any DMA transaction (transmit or receive) is in progress, and if the debug mode is requested while a byte is being transmitted or received, the I^{2}C module enters IPG DEBUG mode after completing the transmission or reception of the current byte. As soon as the module exits IPG DEBUG mode, transmission or reception of the remaining bytes resumes. Refer to the simulation snapshot shown in *Figure 863* for details where the I^{2}C is transmitting 8 bytes using DMA and IPG DEBUG mode is requested while a second byte is being transmitted. IPG DEBUG mode is entered after successfully transmitting the second byte (IPG Debug Halted signal asserted). As soon as the module exits IPG DEBUG mode (IPG Debug Halted signal de-asserted), transmission resumes successfully.

Figure 863. Simulation result of IPG Debug Halted in case of frame transmission using DMA



No more transaction can take place until the debug signal is de-asserted after which the I^{2}C module starts functioning normally. There is a status halted signal IBDBG[IPG_DEBUG_HALTED] to indicate to the user that the I^{2}C module has entered the IPG DEBUG mode. Refer to *Figure 864* for more details.

Figure 864. PG DEBUG mode

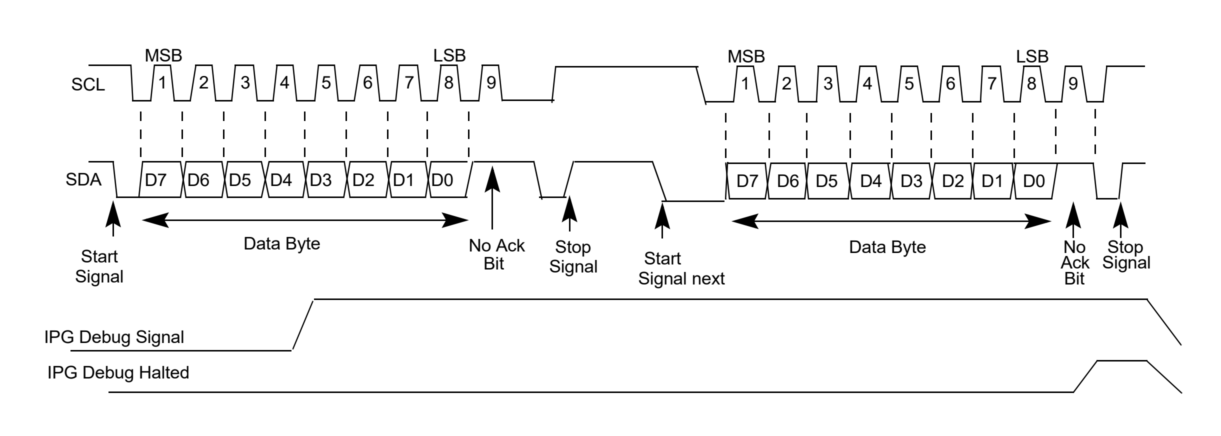
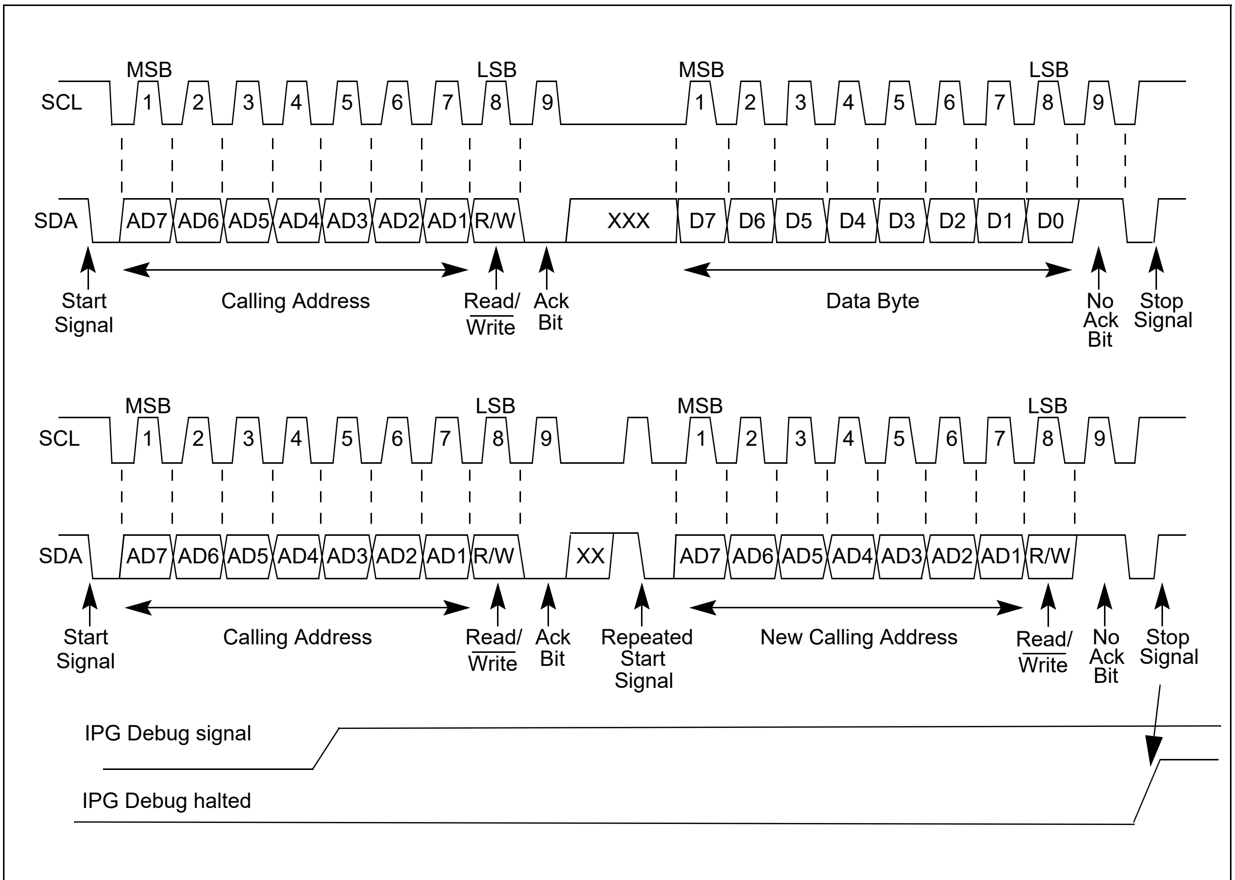


Figure 865 shows a case of IPG DEBUG mode with repeated START transaction. In this case, if the debug signal is asserted in between the transaction, the entire transaction with multiple repeated start will be completed before the I^2C module enters IPG DEBUG mode.

Figure 865. IPG debug mode with repeated start



48.5.8 DMA interface

A simple DMA interface is implemented so that the I^2C can request data transfers with minimal support from the CPU (refer to *Section 48.6.5*). DMA mode is enabled by setting bit 1 in the Control Register (IBCR).

The DMA interface is operational when the I^2C module is configured for Master mode.

At least three bytes of data per frame must be transferred from/to the slave when using DMA mode, although in practice it will only be worthwhile using the DMA mode when there is a large number of data bytes to transfer per frame.

Two internal signals, TX request and RX request, are used to signal the DMA controller when the I^2C module requires data to be written or read from the data register.

Further details of the DMA interface can be found in the *Section 48.6*.

48.6 Initialization and application information

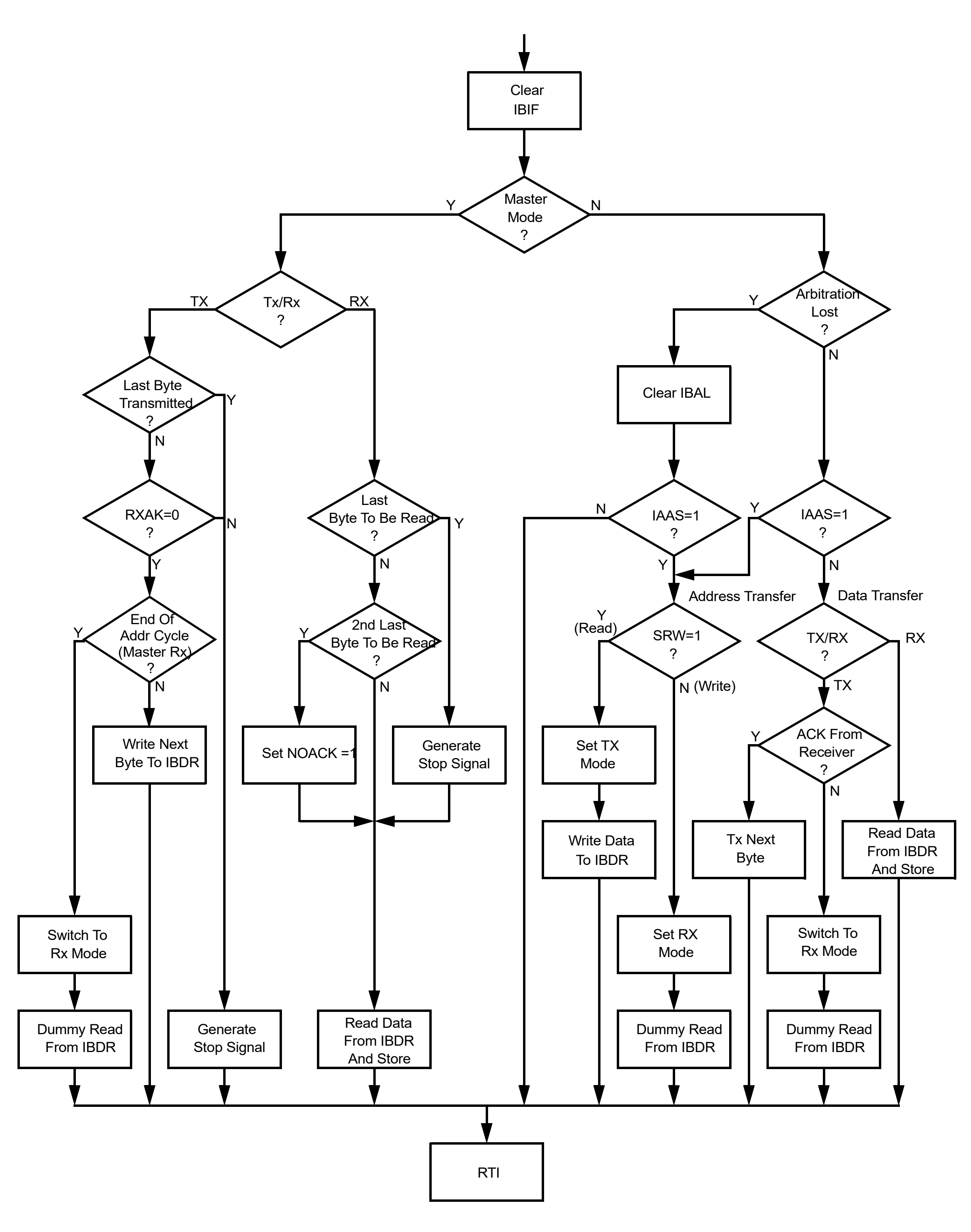
This section presents the following topics:

- *Section 48.6.1: Recommended interrupt service flow*
- *Section 48.6.2: General programming guidelines (for both master and slave mode)*
- *Section 48.6.3: Programming guidelines specific to master mode*
- *Section 48.6.4: Programming guidelines specific to slave mode*
- *Section 48.6.5: DMA application information*

48.6.1 Recommended interrupt service flow

Figure 866 shows a flowchart for the recommended I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C bus behavior.

Figure 866. Recommended I²C interrupt service routine flowchart



48.6.2 General programming guidelines (for both master and slave mode)

This section provides programming guidelines recommended for the I^2C module in both master and slave mode. It presents the following topics:

- *Section 48.6.2.1: Initializing the I2C module*
- *Section 48.6.2.2: Software response after a transfer*

48.6.2.1 Initializing the I^2C module

The following sequence initializes the I^2C module:

1. Use the IBFD register to select the required division ratio to obtain SCL frequency from system clock.
2. Use the IBAD register to define the slave address.
3. Clear the IBCR[IBDIS] bit to enable the I^2C interface system.
4. Use the IBCR to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not.
5. (Optional) Use the IBIC register to further refine the interrupt behavior.

48.6.2.2 Software response after a transfer

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I^2C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing one (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress whenever data register is written to in transmit mode, or during reading out from data register in receive mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I^2C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I^2C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

When a “Transfer Complete” interrupt occurs at the end of the address cycle, the master will always be in transmit mode, that is the address is transmitted. If master receive mode is required, indicated by R/W bit sent with slave calling address, then the Tx/Rx bit at Master side should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

In slave mode, IAAS bit will get set in IBSR if Slave address (IBAD) matches the Master calling address. This is an indication that Master-Slave data communication can now start. During address cycles (IBSR[IAAS]=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IBSR[IAAS]=0), the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

48.6.3 Programming guidelines specific to master mode

This section presents the following topics:

- *Section 48.6.3.1: Generating START*
- *Section 48.6.3.2: Transmit/receive sequence*
- *Section 48.6.3.3: Generating STOP*
- *Section 48.6.3.4: Generating repeated START*
- *Section 48.6.3.5: Loss of arbitration*

48.6.3.1 Generating START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I^2C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB = 0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (that is the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^2C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (IBSR[IBB]==1)                    // wait in loop for IBB flag to clear
IBCR[MS/MSL] and IBCR[]Tx/Rx] = 1       // master and transmit mode, that is,
                                        // generate start condition
IBDR = calling_address                  // send the calling address to the data
register
while (bit 5, IBSR ==0)                 // wait in loop for IBB flag to be set
```

48.6.3.2 Transmit/receive sequence

The following tables present the sequences for:

- Master transmit
- Master receive
- Slave transmit
- Slave receive

Table 951. Master transmit sequence

Step	Action
a	Use the IBFD register to select the required division ratio to obtain SCL frequency from Platform clock/2.
b	Write 0 to IBCR[IBDIS] to enable the I^2C interface system.
c	Use the IBCR to select Master mode, Transmit mode and interrupt enable.
d	Write 0 to IBSR[IBIF].

Table 951. Master transmit sequence (continued)

Step	Action
e	Write data to IBDR.
f	Observe changes in IBSR[TCF]: – When IBSR[TCF] becomes 0, the transfer is in progress. – When IBSR[TCF] becomes 1, the transfer is complete.
g	Wait until IBSR[IBIF] = 1.
h	Read the fields in the IBSR to determine what happened: – If TCF = 1, the transfer completed. – If RXAK = 1, a No Acknowledge condition occurred. – If IBB = 0, the bus transitioned from Busy to Idle state. – If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **Note:** Address Detect (IAAS = 1) can be ignored for master mode (it is valid only for slave mode).
i	Examine IBSR[RXAK] for an acknowledgment from the slave.
j	Repeat steps d through i to transfer the next consecutive bytes of data.

Table 952. Master receive sequence

Step	Action
a	Follow steps a through i in *Table 951* for address dispatch.
b	Write 0 to IBSR[IBIF].
c	Write 0 to IBCR[TXRX] to select Receive mode.
d	Perform a dummy read of the IBDR to initiate the receive operation.
e	Wait until IBSR[TCF] becomes 1. (This proves that the transfer is complete.)
f	Wait until IBSR[IBIF] = 1.
g	Read the fields in the IBSR to determine what happened: – If TCF = 1, the reception completed. – If IBB = 0, the bus transitioned from Busy to Idle state. – If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **Note:** The No Acknowledge condition (RXAK = 1) can be ignored in receive mode.
h	Read the IBDR to determine the data received from the slave.

Table 953. Slave transmit sequence

Step	Action
a	Write 0 to IBCR[IBDIS] to enable the I2C interface system.
b	Examine fields in the IBSR as follows: – If IAAS = 1, examine IBSR[SRW]. – If IAAS = 1 and SRW = 1, write 1 to IBCR[TXRX] to select Transmit mode.
c	Write data to IBDR.
d	Wait until IBSR[IBIF] = 1.

Table 953. Slave transmit sequence (continued)

Step	Action
e	Wait until IBSR[RXAK] = 0.
f	Write 0 to IBSR[IBIF].
g	Repeat steps c through for the next consecutive data transfers.

Table 954. Slave receive sequence

Step	Action
a	Write 0 to IBCR[IBDIS] to enable the I2C interface system.
b	Examine fields in the IBSR as follows: – If IAAS = 1, examine IBSR[SRW]. – If IAAS = 1 and SRW = 0, write 0 to IBCR[TXRX] to select Receive mode.
c	Write 0 to IBSR[IBIF].
d	Perform a dummy read of the IBDR to initiate the receive operation.
e	Wait until IBSR[TCF] becomes 1. (This proves that the transfer is complete.)
f	Wait until IBSR[IBIF] = 1.
g	Read the fields in the IBSR to determine what happened: – If TCF = 1, the reception completed. – If IBB = 0, the bus transitioned from Busy to Idle state. – If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **Note:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
h	Read the IBDR to determine the data received from the master.

48.6.3.3 Generating STOP

A data transfer ends with a STOP signal generated by the “master” device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a STOP condition is generated by a master transmitter.

```
if (tx_count == 0) or          // check to see if all data bytes have been
transmitted
    (bit 0, IBSR == 1) {       // or if no ACK generated
         clear bit 5, IBCR     // generate stop condition
        }
else {
IBDR = data_to_transmit        // write byte of data to DATA register
         tx_count --           // decrement counter
      }                        // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the NOACK bit in IBCR before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --                    // decrease the rx counter
if (rx_count ==1)              // 2nd last byte to be read ?
   bit 3, IBCR = 1             // disable ACK
if (rx_count == 0)             // last byte to be read ?
  bit 5, IBCR = 0              // generate stop signal
else
data_received = IBDR           // read RX data and store
```

48.6.3.4 Generating repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown below.

```
bit 2, IBCR = 1                // generate another start (restart)
IBDR == calling_address        // transmit the calling address
```

48.6.3.5 Loss of arbitration

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL = 1 and MS/SL = 0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU, set the IBAL to indicate that the attempt to engage the bus is failed, and not set the TCF due to the loss of data during arbitration. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

48.6.4 Programming guidelines specific to slave mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an “end of data” signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

48.6.5 DMA application information

The DMA interface on the I^{2}C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the DMA enable bit in the control register is not set when the I^2C module is configured in slave mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I^2C block.

The CPU should also keep the I^2C interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The DMAEN bit in the IBCR register works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, dependant on the setting of the DMAEN bit. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the I^2C should never be configured for slave operation.

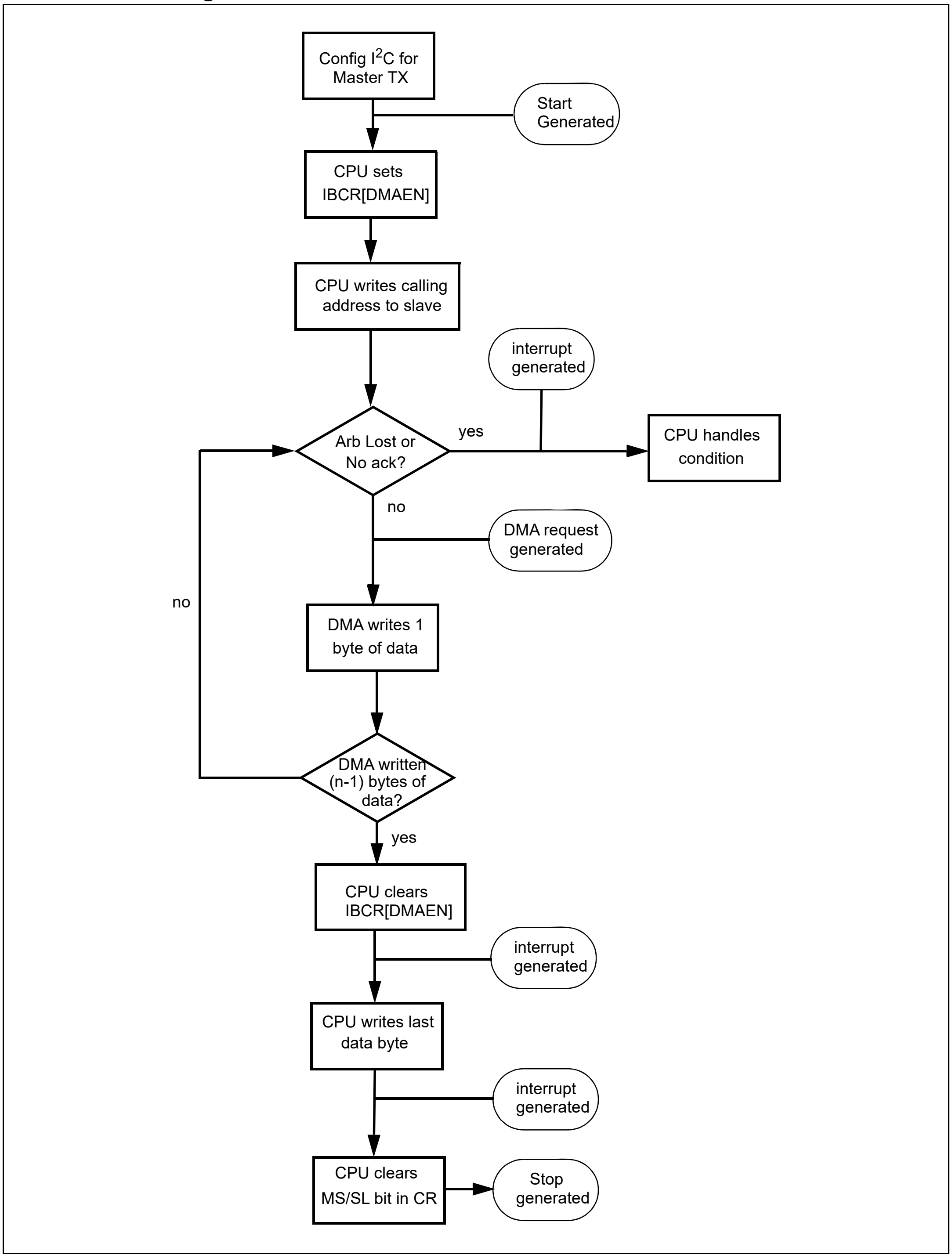
The following sections detail how to set up a DMA transfer and what intervention is required from the CPU. It is assumed that the system DMA controller is capable of generating an interrupt after a certain number of DMA transfers have taken place. The sections present the following topics:

- *Section 48.6.5.1: DMA mode, master transmit*
- *Section 48.6.5.2: DMA mode, master reception*
- *Section 48.6.5.3: Exiting DMA mode, system requirement considerations*

48.6.5.1 DMA mode, master transmit

The following flow diagram details exactly the operation for using a DMA controller to transmit ‘n’ data bytes to a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the last data byte) can be transferred by the DMA controller. The last data byte must be transferred by the CPU.

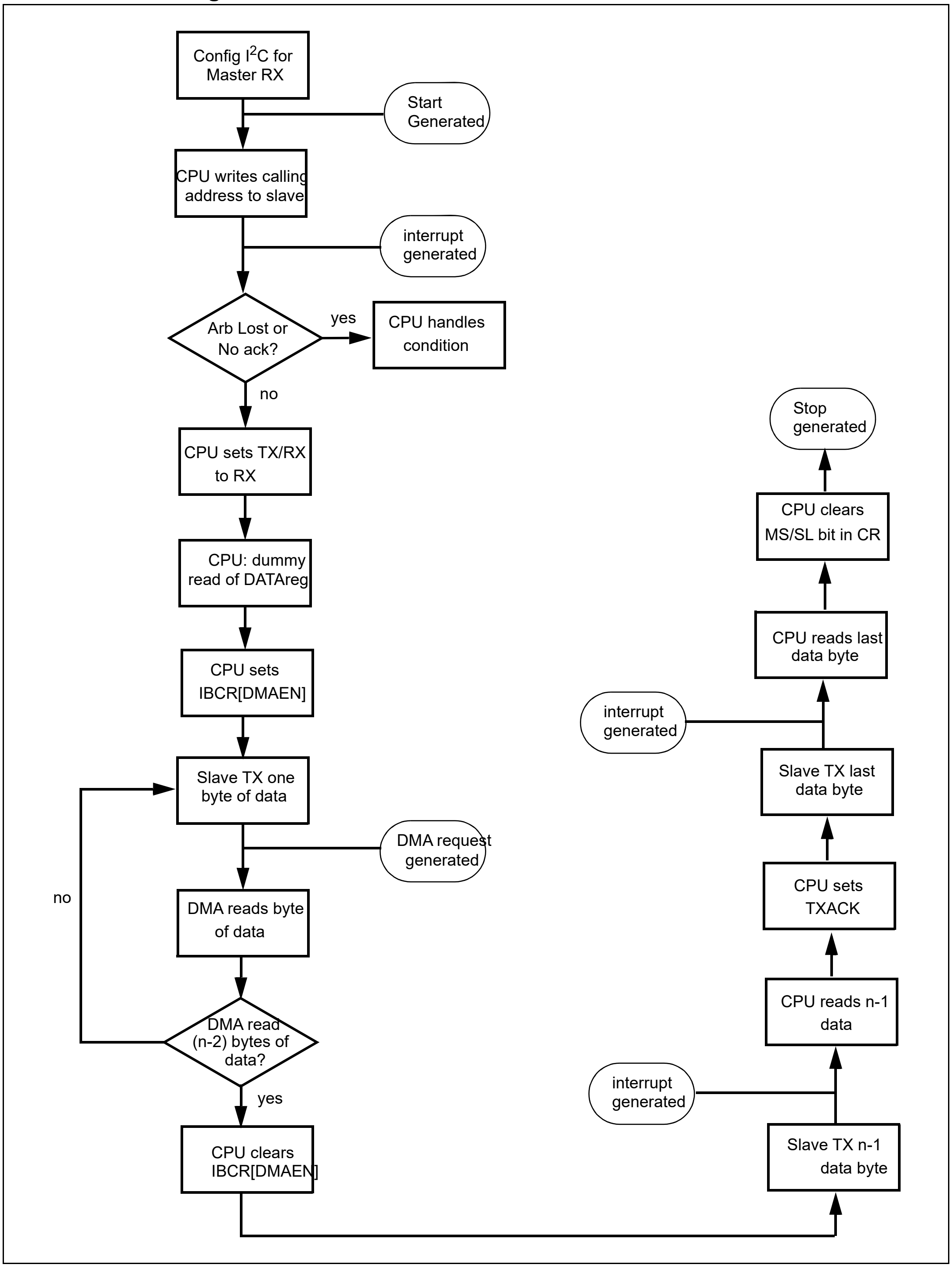
Figure 867. Flow-Chart of DMA mode master transmit



48.6.5.2 DMA mode, master reception

The following flow diagram details the exact operation for using a DMA controller to receive ‘n’ data bytes from a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the two last data bytes) can be read by the DMA controller. The last two data bytes must be transferred by the CPU.

Figure 868. Flow-Chart of DMA mode master receive



48.6.5.3 Exiting DMA mode, system requirement considerations

As described above, the final transfers of both TX and RX transfers need to be handled via interrupt by the CPU. To change from DMA to interrupt driven transfers in the I^{2}C module, you have to disable the DMAEN bit in the IBCR register. The trigger to exit the DMA mode is that the programmed DMA Transfer Control Descriptor (TCD) has completed all its transfers to/from the I^2C module.

After the last DMA write (TX mode) to the I^2C the module will immediately start the next I^2C-bus transfer. The same is true for receive mode. After the DMA read from the IBDR register the module initiates the next I^2C-bus transfer. This results in two possible scenarios in the DMA mode exiting scheme.

1. Fast reaction

   The DMAEN bit is cleared before the next I^2C-bus transfer completes. In this case the module will raise an interrupt request to the CPU which can be serviced normally.

2. Slow reaction

   The DMAEN bit is cleared after the next I^2C-bus transfer has already completed. In this case, the module will not raise an interrupt request to the CPU. Instead the TCF bit can be read to determine that the transfer completed and the module is ready for further transfer.

What is fast/slow reaction?

The reaction time T_R for the system to disable DMAEN after the last DMA controller access to the I^2C is the time required for one byte transfer over the I^2C. For “fast reaction” the disabling has to occur before the 9^{th} bit of the data transfer which is the ACK bit. So the time available is eight times the SCL period.

Equation 36 $T_R = 8 \times T_{SCL}$

In fast mode, with 400 kbit/s, T_{SCL} is 2.5 µs, so T_R is 20 µs.

Depending on the system and DMA controller there are different possibilities for the de-assertion of DMAEN. Three options are:

1. CPU intervention via Interrupt

   The DMA controller is programmed to signal an interrupt to the CPU which is then responsible for the de-assertion of DMAEN. This scheme should be supported by most systems but it can result in a slow reaction time if other higher priority interrupts interfere. Therefore the interrupt handling routine can become complicated as it has to check which of the two cases happened (check TCF bit) and act accordingly. In case of slow reaction you can force an interrupt for the I^2C in the interrupt controller to have the further transfer handled by the normal I^2C interrupt routine.

Note: The use of nested interrupts can still cause potential issues in this scenario, if someone tries to stall the DMA interrupt between the de-assertion and DMAEN bit and checks the TCF bit.

2. DMA channel linking

   If the Transfer control descriptor in the DMA controller that performs the data transfer is linked to another channel that does a write to IBCR to disable the DMAEN field, this might probably be the fastest system solution, but it uses two DMA channels.

Note: Make sure, at system level, that no higher priority DMA requests occur between the two linked TCDs as they could create a slow reaction scenario again.

3. DMA scatter/gather process

   If the Transfer control descriptor in the DMA controller that performs the data transfer has the scatter/gather feature activated, this feature will initiate a reload of another TCD from system RAM after the completion of the first TCD. The new TCD will have its start bit already set and immediately start the required write to the IBCR to disable the DMAEN field. This TCD also has scatter/gather activated and is programmed to reload the initial TCD upon completion, bringing the system back into a “ready-for-I^2C-transfer” state. The advantage over the two other solutions is that this requires neither CPU intervention nor a second DMA channel. This comes at the cost of 64 bytes RAM (two TCDs), some system bus transfer overhead and a little increase in application code complexity.

Note: Make sure, at system level, that no higher priority DMA requests occur during the scatter/gather process, as they could create a slow reaction scenario again.

Example latencies for a 32 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

Equation 37 $4 \times T_{IPI} = 4 / 16\ \mathrm{MHz} = 250\ \mathrm{ns}$

- Reloading a new TCD (8 x 32-bit) via AHB to the DMA controller (scatter/gather process):

Equation 38 $8 \times T_{AHB} = 8 / 32\ \mathrm{MHz} = 250\ \mathrm{ns}$

Example latencies for a 150 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I2C could be faster):

Equation 39 $4 \times T_{IPI} = 4/150\ \mathrm{MHz} = 26.6\ \mathrm{ns}$

- Reloading a new TCD (4 x 64-bit) via AHB to the DMA controller (scatter/gather process):

Equation 40 $4 \times T_{AHD} = 4/150\ \mathrm{MHz} = 26.6\ \mathrm{ns}$

With the DMA scatter/gather process the required IBCR access can be done in 0.5 µs, leaving a large margin of 19.5 µs for additional system delays. In this way, the slow reaction case can be prevented. The system user needs to decide which usage model best suits his overall requirement.

49 Deserial Serial Peripheral Interface (DSPI)

49.1 Introduction

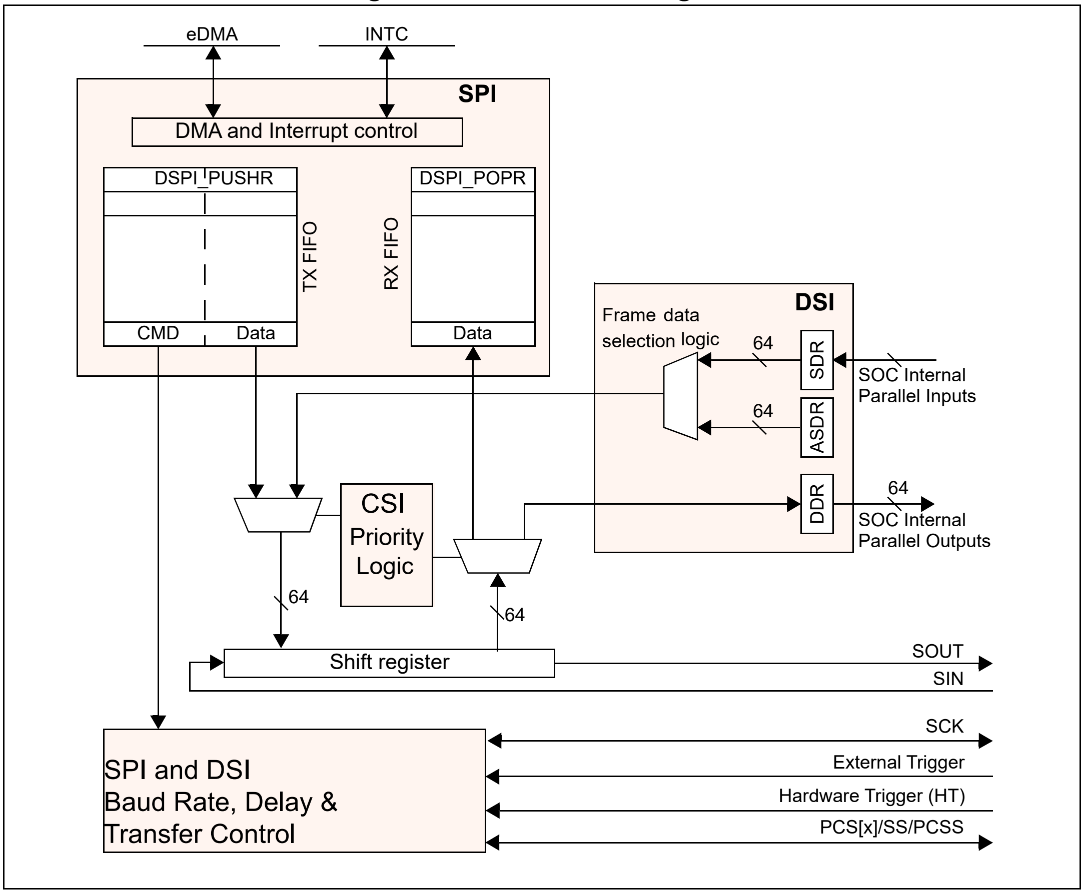
The Deserial Serial Peripheral Interface (DSPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device. The module supports MCU pin count reduction through serialization and deserialization of MCU internal signals transmitted over the SPI serial link.

Note: For the chip-specific implementation details of the instances of this module, refer to the Device Configuration chapter.

49.1.1 Block diagram

Figure 869 provides the block diagram of this module.

Figure 869. DSPI block diagram



49.1.2 Features

The DSPI supports the following features:

- Full-duplex, four-wire synchronous transfers.
- Master and Slave modes: data streaming operation in slave mode with continuous slave selection.
- Buffered transmit operation with 4-entry TX FIFO buffer.
- Buffered receive operation with 4-entry RX FIFO buffer.
- Asynchronous clocking scheme for Register and Protocol interfaces.
- Individual TX and RX FIFOs disabling for low-latency updates to SPI queues.
- Visibility inside TX and RX FIFOs for debugging.
- Programmable transfer attributes on a per-frame basis:
  - 8 transfer attribute registers along with 8 extended transfer attribute registers
  - Serial clock with programmable polarity and phase
  - Various programmable delays
  - programmable serial frame size of 4 to 64 bits, expandable by software control. SPI frames longer than 32 bits are supported using the continuous selection format
  - Continuously held chip select capability
  - Parity control
- 8 Peripheral Chip Selects, expandable to 256 with external demultiplexer.
- Deglitching support for up to 128 Peripheral Chip Selects with external demultiplexer.
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
  - CMD FIFO is not full (CMDFFF)
- Interrupt conditions:
  - End Of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - CMD FIFO is not full (CMDFFF)
  - Transfer of current frame Complete (TCF)
  - Transfers due from current Command frame Complete (CMDTCF)
  - Transfer of current SPI frame Complete (SPITCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - Frame received while Receive FIFO is full (RFOF)
  - SPI Parity Error (SPEF)
  - Data present in TX FIFO while CMD FIFO is empty (TFIWF)
- Global interrupt request line.
- Modified SPI transfer formats for communication with slower peripheral devices.
- Power-saving architectural features: support for stop mode

The DSPI also supports pin reduction through serialization and deserialization if enabled for the module.

- Two sources of serialized data:
  - DSPI memory-mapped register
  - Parallel input signals
  - Programmable selection of source data on bit basis.
- Deserialized data provided as:
  - Parallel output signals
  - Bits in a memory-mapped register
- Interrupt conditions:
  - Deserialized data matches preprogrammed pattern (DDIF)
  - Transfer of current DSI frame complete (DSITCF)
  - DSI parity error (DPEF)
- DMA request support for following conditions:
  - Deserialized data matches preprogrammed pattern (DDIF)
- Transfer initiation conditions:
  - Continuous
  - Edge sensitive hardware trigger
  - Change in data
- Pin serialization/deserialization with interleaved SPI frames for control and diagnostics.
- Continuous serial communications clock.

DSPI supports a combination of SPI and DSI modes of operation (Combined Serial Interface (CSI)) for the downstream Micro Second Channel in either Timed Serial Bus (TSB) configuration or Interleaved Frames Configuration (configurable by software). Both TSB and Interleaved TSB modes have the following common features:

- Transmission of frames is performed at frame boundaries.
- Frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame.
- Separate interrupts for frame completion from SPI and DSI.

49.1.3 DSPI configurations

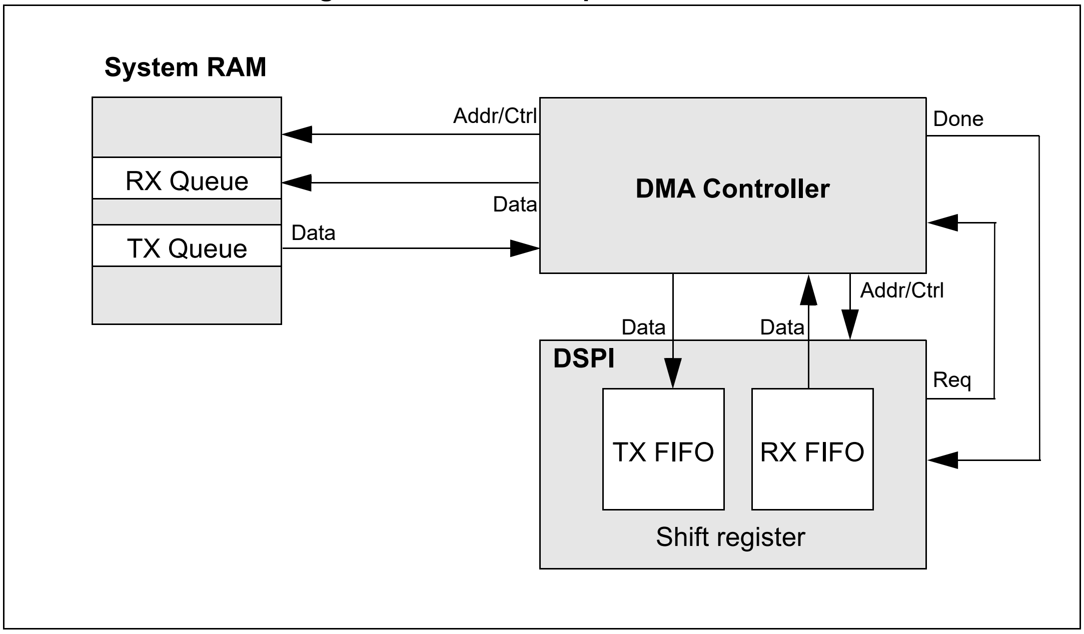
The DSPI module can operate in SPI configuration as well as in DSI, and CSI configuration. Refer to the relevant section in the “Communication interfaces” section of the “Device Configuration” chapter.

49.1.3.1 SPI configuration

The SPI configuration allows the DSPI to send and receive serial data. This configuration allows the DSPI to operate as a basic SPI block with internal FIFOs supporting external queues operation. Transmit data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the receive FIFO and write transmit data to the transmit FIFO.

For queued operations, the SPI queues can reside in system RAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished by a DMA controller or host CPU. *Figure 870* shows a system example with DMA, DSPI and external queues in system RAM.

Figure 870. DSPI with queues and DMA



49.1.3.2 DSI configuration

In the DSI configuration the DSPI serializes up to 64 parallel input signals or register bits. The DSPI also deserializes the received data to parallel output signals or to a memory-mapped register. The data is transferred using a SPI-like protocol.

TSB mode provides the Micro Second downstream Channel support (MSC), serializing from 4 to 64 parallel input signals or register bits. Refer to *Section 49.4.9: Timed Serial Bus (TSB)*.

49.1.3.3 CSI configuration

There are three configurations available in this mode.

- The normal CSI configuration is a combination of the SPI and DSI configurations. In this configuration the DSPI interleaves DSI data frames with SPI data frames. Interleaving is done on the frame boundaries.
- In the TSB configuration, transmission of SPI data has higher priority than DSI data.
- In the Interleaved TSB (ITSB) configuration, the frames from SPI and DSI are interleaved without priority. On every trigger, frames from DSI are sent when there are no frames in the SPI or the previous transmission was a frame from SPI. ITSB is detailed in *Section 49.4.10: Interleaved TSB (ITSB) Mode*.

49.1.4 Modes of operation

The DSPI modes of operation can be divided into two categories:

- Module-specific modes
  - Master mode
  - Slave mode
  - Module Disable mode
- MCU-specific modes
  - External Stop mode
  - Debug mode

The DSPI enters module-specific modes when the host writes a DSPI register. The MCU-specific modes are controlled by signals external to the DSPI. The MCU-specific modes are modes that an MCU may enter in parallel to the DSPI block-specific modes.

49.1.4.1 Master mode

Master mode allows the DSPI to initiate and control serial communication. In this mode, the SCK signal and the PCS[x] signals are controlled by the DSPI and configured as outputs.

49.1.4.2 Slave mode

Slave mode allows the DSPI to communicate with SPI/DSI bus masters. In this mode, the DSPI responds to externally controlled serial transfers. The SCK signal and the PCS[0]/SS signals are configured as inputs and driven by a SPI bus master.

49.1.4.3 Module Disable mode

The Module Disable mode can be used for MCU power management. The clock to the non-memory-mapped logic in the DSPI can be stopped while in Module Disable mode.

49.1.4.4 External Stop mode

External Stop mode is used for MCU power management; the DSPI supports the Peripheral Bus Stop mode mechanism.

Upon a request to enter External Stop mode:

- In Master mode, the DSPI block acknowledges the request and completes the transfer in progress. When the DSPI reaches the frame boundary, it signals that the protocol clock to the DSPI module may be shut off.
- In Slave mode, a request to enter External Stop mode must be run after the last clock cycle or Low-power mode is not entered.

49.1.4.5 Debug mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls DSPI behavior in the debug mode. If the bit is set, the DSPI stops all serial transfers, when the MCU is in Debug mode. If the bit is cleared, the MCU Debug mode has no effect on the DSPI.

49.2 DSPI signal description

This section provides the DSPI signals description.

Table 955 lists the signals that may connect off chip depending on device implementation.

Table 955. DSPI signal description

Signal	Description	I/O
PCS0/ SS	Master mode: Peripheral Chip Select 0 output Slave mode: Slave Select input	I/O
PCS[3:1]	Master mode: Peripheral Chip Select 1–3 Slave mode: Unused	O
PCS4	Master mode: Peripheral Chip Select 4	O
PCS5/ PCSS	Master mode: Peripheral Chip Select 5 / Peripheral Chip Select Strobe Slave mode: Unused	O
PCS[7:6]	Master mode: Peripheral Chip Select 6–7 Slave mode: Unused	O
SIN	Serial Data In	I
SOUT	Serial Data Out	O
SCK	Master mode: Serial Clock (output) Slave mode: Serial Clock (input)	I/O
HT	Hardware Trigger	I

49.2.1 PCS0/$\overline{SS}$ — Peripheral Chip Select/Slave Select

In master mode, the PCS0 signal is a Peripheral Chip Select output that selects which slave device the current transmission is intended for.

In slave mode, the active low $\overline{SS}$ signal is a Slave Select input signal that allows a SPI master to select the DSPI as the target for transmission.

49.2.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3

PCS1 – PCS3 are Peripheral Chip Select output signals in master mode.

In slave mode, these signals are unused.

49.2.3 PCS4 — Peripheral Chip Select 4

In master mode, PCS4 is a Peripheral Chip Select output signal.

49.2.4 PCS5/$\overline{PCSS}$ — Peripheral Chip Select 5/Peripheral Chip Select Strobe

PCS5 is a Peripheral Chip Select output signal. When the DSPI is in master mode and the MCR[PCSSE] bit is cleared, this signal selects which slave device the current transfer is intended for.

When the DSPI is in master mode and the MCR[PCSSE] bit is set, the $\overline{\text{PCSS}}$ signal acts as a strobe to an external peripheral chip select demultiplexer, which decodes the PCS0 – PCS4 and PCS6 – PCS7 signals, preventing glitches on the demultiplexer outputs.

This signal is not used in slave mode.

49.2.5 PCS[6] – PCS[7] — Peripheral Chip Selects 6 – 7

PCS[6] – PCS[7] are Peripheral Chip Select output signals in master mode. In slave mode, these signals are not used.

49.2.6 SIN — Serial Input

SIN is a serial data input signal.

49.2.7 SOUT — Serial Output

SOUT is a serial data output signal.

49.2.8 SCK — Serial Clock

SCK is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK is an input from an external bus master.

49.2.9 HT — Hardware Trigger

In master mode while in DSI or CSI Configurations, the HT signal initiates a data transfer when the DSICR0[TRRE] bit is set and a rising or falling edge is detected on HT. Which edge to trigger on is determined by the DSICR0[TPOL] bit.

49.3 Memory map/register definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the DSPI_POPR also results in a transfer error.

Table 956. DSPI memory map

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_0000	DSPI Module Configuration Register (DSPI_MCR)	32	R/W	0000_4001h	*Section 49.3.1*
0000_0004	Hardware Configuration Register (DSPI_HCR)	32	R	0133_3000h	*Section 49.3.2*
0000_0008	DSPI Transfer Count Register (DSPI_TCR)	32	R/W	0000_0000h	*Section 49.3.3*
0000_000C	DSPI Clock and Transfer Attributes Register 0 (In Master Mode) (DSPI_CTAR0)	32	R/W	7800_0000h	*Section 49.3.4*
0000_000C	DSPI Clock and Transfer Attributes Register 0 (In Slave Mode) (DSPI_CTAR0_SLAVE)	32	R/W	7800_0000h	*Section 49.3.5*
0000_0010	DSPI Clock and Transfer Attributes Register 1 (In Master Mode) (DSPI_CTAR1)	32	R/W	7800_0000h	*Section 49.3.4*

Table 956. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_0010	DSPI Clock and Transfer Attributes Register 1 (In Slave Mode) (DSPI_CTAR1_SLAVE)	32	R/W	7800_0000h	Section 49.3.5
0000_0014	DSPI Clock and Transfer Attributes Register 2 (In Master Mode) (DSPI_CTAR2)	32	R/W	7800_0000h	Section 49.3.4
0000_0018	DSPI Clock and Transfer Attributes Register 3 (In Master Mode) (DSPI_CTAR3)	32	R/W	7800_0000h	Section 49.3.4
0000_001C	DSPI Clock and Transfer Attributes Register 4 (In Master Mode) (DSPI_CTAR4)	32	R/W	7800_0000h	Section 49.3.4
0000_0020	DSPI Clock and Transfer Attributes Register 5 (In Master Mode) (DSPI_CTAR5)	32	R/W	7800_0000h	Section 49.3.4
0000_0024	DSPI Clock and Transfer Attributes Register 6 (In Master Mode) (DSPI_CTAR6)	32	R/W	7800_0000h	Section 49.3.4
0000_0028	DSPI Clock and Transfer Attributes Register 7 (In Master Mode) (DSPI_CTAR7)	32	R/W	7800_0000h	Section 49.3.4
0000_002C	DSPI Status Register (DSPI_SR)	32	R/W	Refer to the register description	Section 49.3.6
0000_0030	DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)	32	R/W	0000_0000h	Section 49.3.7
0000_0034	DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)	32	R/W	0000_0000h	Section 49.3.8
0000_0034	DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)	32	R/W	0000_0000h	Section 49.3.9
0000_0038	DSPI POP FIFO Register (DSPI_POPR)	32	R	0000_0000h	Section 49.3.10
0000_003C	DSPI Transmit FIFO Register 0 (DSPI_TXFR0)	32	R	0000_0000h	Section 49.3.11
0000_0040	DSPI Transmit FIFO Register 1 (DSPI_TXFR1)	32	R	0000_0000h	Section 49.3.11
0000_0044	DSPI Transmit FIFO Register 2 (DSPI_TXFR2)	32	R	0000_0000h	Section 49.3.11
0000_0048	DSPI Transmit FIFO Register 3 (DSPI_TXFR3)	32	R	0000_0000h	Section 49.3.11
0000_007C	DSPI Receive FIFO Register 0 (DSPI_RXFR0)	32	R	0000_0000h	Section 49.3.12
0000_0080	DSPI Receive FIFO Register 1 (DSPI_RXFR1)	32	R	0000_0000h	Section 49.3.12
0000_0084	DSPI Receive FIFO Register 2 (DSPI_RXFR2)	32	R	0000_0000h	Section 49.3.12
0000_0088	DSPI Receive FIFO Register 3 (DSPI_RXFR3)	32	R	0000_0000h	Section 49.3.12

Table 956. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_00BC	DSPI DSI Configuration Register 0 (DSPI_DSICR0)	32	R/W	0000_0000h	Section 49.3.13
0000_00C0	DSPI DSI Serialization Data Register 0 (DSPI_SDR0)	32	R	0000_0000h	Section 49.3.14
0000_00C4	DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)	32	R/W	0000_0000h	Section 49.3.15
0000_00C8	DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)	32	R	0000_0000h	Section 49.3.16
0000_00CC	DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)	32	R	0000_0000h	Section 49.3.17
0000_00D0	DSPI DSI Configuration Register 1 (DSPI_DSICR1)	32	R/W	0000_0000h	Section 49.3.18
0000_00D4	DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)	32	R/W	0000_0000h	Section 49.3.19
0000_00D8	DSPI DSI Parallel Input Select Register 0 (DSPI_PISR0)	32	R/W	0000_0000h	Section 49.3.20
0000_00DC	DSPI DSI Parallel Input Select Register 1 (DSPI_PISR1)	32	R/W	0000_0000h	Section 49.3.21
0000_00E0	DSPI DSI Parallel Input Select Register 2 (DSPI_PISR2)	32	R/W	0000_0000h	Section 49.3.22
0000_00E4	DSPI DSI Parallel Input Select Register 3 (DSPI_PISR3)	32	R/W	0000_0000h	Section 49.3.23
0000_00E8	DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)	32	R/W	0000_0000h	Section 49.3.24
0000_00EC	DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)	32	R/W	0000_0000h	Section 49.3.25
0000_00F0	DSPI DSI Serialization Data Register 1 (DSPI_SDR1)	32	R	0000_0000h	Section 49.3.26
0000_00F4	DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)	32	R/W	0000_0000h	Section 49.3.27
0000_00F8	DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)	32	R	0000_0000h	Section 49.3.28
0000_00FC	DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)	32	R	0000_0000h	Section 49.3.29
0000_0100	DSPI DSI Serialization Source Select Register 1 (DSPI_SSR1)	32	R/W	0000_0000h	Section 49.3.30
0000_0104	DSPI DSI Parallel Input Select Register 4 (DSPI_PISR4)	32	R/W	0000_0000h	Section 49.3.31
0000_0108	DSPI DSI Parallel Input Select Register 5 (DSPI_PISR5)	32	R/W	0000_0000h	Section 49.3.32

Table 956. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_010C	DSPI DSI Parallel Input Select Register 6 (DSPI_PISR6)	32	R/W	0000_0000h	Section 49.3.33
0000_0110	DSPI DSI Parallel Input Select Register 7 (DSPI_PISR7)	32	R/W	0000_0000h	Section 49.3.34
0000_0114	DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)	32	R/W	0000_0000h	Section 49.3.35
0000_0118	DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)	32	R/W	0000_0000h	Section 49.3.36
0000_011C	DSPI Clock and Transfer Attributes Register Extended 0 (DSPI_CTARE0)	32	R/W	0000_0001h	Section 49.3.37
0000_0120	DSPI Clock and Transfer Attributes Register Extended 1 (DSPI_CTARE1)	32	R/W	0000_0001h	Section 49.3.37
0000_0124	DSPI Clock and Transfer Attributes Register Extended 2 (DSPI_CTARE2)	32	R/W	0000_0001h	Section 49.3.37
0000_0128	DSPI Clock and Transfer Attributes Register Extended 3 (DSPI_CTARE3)	32	R/W	0000_0001h	Section 49.3.37
0000_012C	DSPI Clock and Transfer Attributes Register Extended 4 (DSPI_CTARE4)	32	R/W	0000_0001h	Section 49.3.37
0000_0130	DSPI Clock and Transfer Attributes Register Extended 5 (DSPI_CTARE5)	32	R/W	0000_0001h	Section 49.3.37
0000_0134	DSPI Clock and Transfer Attributes Register Extended 6 (DSPI_CTARE6)	32	R/W	0000_0001h	Section 49.3.37
0000_0138	DSPI Clock and Transfer Attributes Register Extended 7 (DSPI_CTARE7)	32	R/W	0000_0001h	Section 49.3.37
0000_013C	DSPI Status Register Extended (DSPI_SREX)	32	R	0000_0000h	Section 49.3.38

49.3.1 DSPI Module Configuration Register (DSPI_MCR)

The DSPI_MCR contains bits to configure various attributes associated with DSPI operations. The HALT and MDIS bits can be changed at any time, but they only take effect on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed while the DSPI is in the Running state.

Address: 0x0000 Access: User read/write

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MSTR	CONT_SCKE	DCONF		FRZ	MTFE	PCSSE	ROOE	PCSIS7	PCSIS6	PCSIS5	PCSIS4	PCSIS3	PCSIS2	PCSIS1	PCSIS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0	0			0	0	0	0				
W		MDIS	DIS_TXF	DIS_RXF	CLR_TXF	CLR_RXF	SMPL_PT						XSPI	FCPCS	PES	HALT
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 871. DSPI Module Configuration Register (DSPI_MCR)

Table 957. DSPI_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Configures the DSPI for either master mode or slave mode. 0 DSPI is in slave mode. 1 DSPI is in master mode.
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0 Continuous SCK disabled. 1 Continuous SCK enabled.
2:3 DCONF	DSPI Configuration Selects the DSPI configuration. 00 SPI 01 DSI 10 CSI 11 Reserved
4 FRZ	Freeze Enables the DSPI transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in debug mode. 1 Halt serial transfers in debug mode.
5 MTFE	Modified Timing Format Enable Enables a modified transfer format to be used. MTFE should not be set in ITSB Mode as Microsecond Channel upstream operation is not supported. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.

Table 957. DSPI_MCR field descriptions (continued)

Field	Description
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS[5]/ $\overline{PCSS}$ to operate as a PCS Strobe output signal. 0 PCS[5]/PCSS is used as the Peripheral Chip Select[5] signal (PCS[5]). 1 PCS[5]/PCSS is used as an active-low PCS Strobe signal (PCSS).
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the DSPI to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer generating the overflow is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8:15 PCSIS[7:0]	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory-mapped logic in the DSPI, putting the DSPI in a software controlled power-saving state. The reset value of the MDIS bit is 1. MDIS should be set to ‘0’ in Slave Mode as a slave does not control master transactions. 0 Enable DSPI clocks. 1 Allow external logic to disable DSPI clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the DSPI (TXFIFO and CMD FIFO) operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Tx FIFO is enabled. 1 Tx FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Rx FIFO is enabled. 1 Rx FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes TX FIFO and CMD FIFO. Writing a ‘1’ to CLR_TXF clears the TX FIFO and CMD FIFO Counters. The CLR_TXF bit is always read as zero. 0 Do not clear the Tx FIFO and CMD FIFO counters. 1 Clears the Tx FIFO and CMD FIFO counters.

Table 957. DSPI_MCR field descriptions (continued)

Field	Description
21 CLR_RXF	Clear RX FIFO Flushes RX FIFO. Writing a ‘1’ to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the Rx FIFO counter. 1 Clear the Rx FIFO counter.
22:23 SMPL_PT	Sample Point Controls when the DSPI master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTAR is 0. 00 0 protocol clocks between SCK edge and SIN sample 01 1 protocol clock between SCK edge and SIN sample 10 2 protocol clocks between SCK edge and SIN sample 11 Reserved
28 XSPI	Extended SPI Mode Enables use of CTARE (Command and Transfer Attribute Register Extended) registers, which allow the user to send up to 32-bit SPI frames. Command Cycling is also enabled, allowing the user to send multiple Data Frames in a single Command Frame. When MCR[DIS_TXF] is asserted, Extended SPI Mode cannot be used to transmit SPI frames over 16 bits in size. 0 Normal SPI Mode. Up to 16-bit Frames. Command Cycling is not available. 1 Extended SPI Mode. Up to 32-bit SPI Frames. Command Cycling Enabled.
29 FCPCS	Fast Continuous PCS Mode. This bit enables the masking of “After SCK (t_{ASC})” and “PCS to SCK (t_{CSC})” delays when operating in Continuous PCS mode. The individual delay masks are selected via PUSHR[PE_MASC] and PUSHR{PP_MCSC]. The firmware should select appropriate masks when providing continuous frames via the PUSHR. This masking is not available if Continuous SCK mode is enabled. 0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in PUSHR.
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt Starts and stops DSPI transfers. 0 Start transfers. 1 Stop transfers. **Note:** HALT bit starts/stops the internal state machine. All the other MCR bits must be set to their expected value before releasing HALT bit. If one MCR bit needs to be changed in running state, HALT bit must be set first.

49.3.2 Hardware Configuration Register (DSPI_HCR)

Hardware Configuration Register provides particular implementation details about the module, that is number of Receive and Transmit FIFO entries, number of CTAR registers and if DSI features are implemented in the module or not. It is read only register. A write to this register will generate a Transfer Error.

Note: The reset bits in the HCR are set by the configuration parameters.

Address: 0x0004 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSI	PISR	DSI64	PISR_EX	XSPI	CTAR			TXFR				RXFR			
W																
Reset	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDFR				MSTR_MODE	SLV_MODE	TSB_EN	ITSB_EN	0	0	0	0	0	0	0	0
W																
Reset	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Figure 872. Hardware Configuration Register (DSPI_HCR)

Table 958. DSPI_HCR field descriptions

Field	Description
0 DSI	DSI features are implemented for the module. 0 DSI features are not implemented, DSI registers do not exist. 1 DSI features are implemented
1 PISR	PISR0-3 and parallel inputs frame positions selection logic (DSI Muxing Logic) are implemented for the module. 0 PISR0-3 registers are not implemented. 1 PISR0-3 registers are implemented.
2 DSI64	DSI features in 64 bit mode are implemented for the module. 0 DSI features for 64 bit mode are not implemented; related registers do not exist. 1 DSI features for 64 bit mode are implemented
3 PISR_EX	PISR4-7 and parallel inputs frame positions selection logic (DSI 64 bit Muxing Logic) are implemented for the module. 0 PISR4-7 registers are not implemented. 1 PISR4-7 registers are implemented.
4 XSPI	Extended SPI Mode feature is implemented for the module. 0 Extended SPI Mode is not implemented. 1 Extended SPI Mode is implemented.
5:7 CTAR	Maximum implemented CTAR register number.
8:11 TXFR	Maximum implemented TXFR register number.
12:15 RXFR	Maximum implemented RXFR register number.

Table 958. DSPI_HCR field descriptions (continued)

Field	Description
16:19 CMDFR	Maximum implemented command FIFO number.
20 MSTR_MODE	Master Mode implementation is enabled
21 SLV_MODE	Slave Mode implementation is enabled
22 TSB_EN	TSB Mode implementation is enabled
23 ITSB_EN	Interleaved TSB Mode implementation is enabled

49.3.3 DSPI Transfer Count Register (DSPI_TCR)

DSPI_TCR has an SPI transfer counter to assist in queue management.

Do not write the TCR when the DSPI is in the Running state.

Address: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SPI_TCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 873. DSPI Transfer Count Register (DSPI_TCR)

Table 959. DSPI_TCR field descriptions

Field	Description
0:15 SPI_TCNT	SPI Transfer Counter The SPI_TCNT field increments every time the last bit of a SPI frame is transmitted. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. A value written to SPI_TCNT presets the counter to that value. The Transfer Counter wraps around; incrementing past 65535 resets the counter to zero.

49.3.4 DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

CTARs are used to define different transfer attributes:

- In master mode, the CTARs define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays.
- In slave mode, a subset of the bitfields in CTAR0 and CTAR1 set the slave transfer attributes.

Do not write to the CTARs while the DSPI is in the Running state.

When the DSPI is configured as:

- SPI master – the CTAS field in the command portion of the TX FIFO entry selects which CTAR is used.
- SPI bus slave – the CTAR0 register is used.
- DSI master – the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0) selects which CTAR is used.
- DSI bus slave – the CTAR1 register is used.

In CSI Configuration, the transfer attributes are based on whether the current frame is:

- SPI data – follow the protocol described for SPI Configuration,
- DSI data – follow the protocol described for DSI Configuration.

CSI Configuration is valid only in master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks via PDT and DT fields.

Address: 0x000C + *n**0x4 (*n*=0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 874. DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

Table 960. DSPI_CTARn field descriptions

Field	Description
0 DBR	Double Baud Rate. (Master mode only) Doubles the baud rate of the Serial Communications Clock (SCK). It halves the Baud Rate division ratio, supporting faster frequencies and odd division ratios for the Serial Communications Clock (SCK). When DBR is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in *Table 961*. Refer to the BR field description in this table for details on how to compute the baud rate. Refer to *Table 961* for DSPI SCK duty cycle values. 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
1:4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. The minimum valid value for the number of bits to be transmitted for any mode is 4.There is a constraint on the minimum allowable Frame size, which depends on the register Read/Write clock to the Protocol Clock Ratio. The minimum Frame size (FMSZ + 1) is decided by the following equation. Upper Ceiling must be applied for non-integer values. Min. Frame Size = ((4 x f_p) + (3 x f_r)) / (n x f_r) f_p = Protocol Clock Frequency f_r = Register Read/Write Clock Frequency n = Multiple of protocol clock required to get Baud Clock - Given by (PBR x BR) / (1 + DBR) The minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Note that 'f_p' can be equal to, less than or greater than 'f_r' purely based on user requirement. When the DSPI operates in TSB mode, the FMSZ field value plus 1 is equal to the data frame bit number, where control of the PCS assertion switches from the DSICR0 to the DSICR1 register. If TSB Mode is operated when DSI is used in 64-bit mode, then DSICR0[FMSZ4] also comes into picture while switching PCS assertion control.
5 CPOL	Clock Polarity. (Master and Slave mode) Selects the inactive state of the Serial Communications Clock (SCK). Devices must have identical clock polarities for successful communication between serial devices. When the Continuous Selection Format is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.

Table 960. DSPI_CTARn field descriptions (continued)

Field	Description
6 CPHA	Clock Phase or TSB mode. (Master and Slave mode) Selects which edge of SCK causes data to change and which edge causes data to be captured. Devices must have identical clock phase settings for successful communication between serial devices. In Continuous SCK or TSB modes, this bit is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First. Specifies whether the LSB or MSB of the frame is transferred first. In TSB mode, this bit should comply with the MSC specification. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8:9 PCSSCK	PCS to SCK Delay Prescaler. Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. Refer to the CSSCK field description in this table for calculating the PCS to SCK Delay. Refer to *Section 49.4.5.2: PCS to SCK delay (tCSC)* for more details. In TSB mode, PCSSCK has no effect. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
10:11 PASC	After SCK Delay Prescaler. Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. Refer to the ASC field description in this table for calculating the After SCK Delay. Refer to *Section 49.4.5.3: After SCK delay (tASC)* for more details. In TSB mode, PASC has no effect. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12:13 PDT	Delay after Transfer Prescaler. (Master mode only) Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In TSB mode, the PDT field defines two MSB bits of the Delay after Transfer. Refer to the DT field description for calculating the Delay after Transfer. Refer to *Section 49.4.5.4: Delay after transfer (tDT)* for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.

Table 960. DSPI_CTARn field descriptions (continued)

Field	Description
14:15 PBR	Baud Rate Prescaler. (Master mode only) Selects the prescaler value for the baud rate. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. Refer to the BR field description in this table for calculating the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
16:19 CSSCK	PCS to SCK Delay Scaler. (Master mode only) Selects the scaler value for the PCS to SCK delay. PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period calculated by: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$ *Table 962* lists the delay scaler values. Refer to *Section 49.4.5.2: PCS to SCK delay (tCSC)* for more details. In TSB mode, the field has no effect.
20:23 ASC	After SCK Delay Scaler. (Master mode only) Selects the scaler value for the After SCK Delay. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period calculated by: $t_{ASC} = (1/f_P) \times PASC \times ASC$ *Table 962* lists the delay scaler values. Refer to *Section 49.4.5.3: After SCK delay (tASC)* for more details.
24:27 DT	Delay After Transfer Scaler (Master mode only) Selects the Delay after Transfer Scaler. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period. The Delay after Transfer is a multiple of the protocol clock period calculated by: $t_{DT} = (1/f_P) \times PDT \times DT$ *Table 962* lists the delay scaler values. In the TSB mode, the Delay after Transfer is equal to a number formed by concatenation of the PDT and DT fields plus 1 of the SCK clock periods. Refer to *Section 49.4.5.2: PCS to SCK delay (tCSC)* for more details.
28:31 BR	Baud Rate Scaler. (Master mode only) Selects the scaler value for the baud rate. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is calculated by: SCK baud rate = $(f_P/PBR) \times [(1+DBR)/BR]$ *Table 963* lists the baud rate scaler values.

Table 961. DSPI SCK duty cycle

DBR	CPHA	PBR	SCK duty cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table 962. Delay scaler encoding

Field value	Delay scaler value
0000	2
0001	4
0010	8
0011	16
0100	32
0101	64
0110	128
0111	256
1000	512
1001	1024
1010	2048
1011	4096
1100	8192
1101	16384
1110	32768
1111	65536

Table 963. DSPI baud rate scaler

CTAR*n*[BR]	Baud rate scaler value
0000	2
0001	4
0010	6

Table 963. DSPI baud rate scaler (continued)

CTAR*n*[BR]	Baud rate scaler value
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

49.3.5 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)

When the DSPI is configured as:

- SPI master – the CTAS field in the command portion of the TX FIFO entry selects which CTAR is used.
- SPI bus slave – the CTAR0 register is used.
- DSI master – the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0) selects which CTAR is used.
- DSI bus slave – the CTAR1 register is used.

In CSI Configuration, the transfer attributes are based on whether the current frame is:

- SPI data – follow the protocol described for SPI Configuration,
- DSI data – follow the protocol described for DSI Configuration.

CSI Configuration is valid only in master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks via PDT and DT fields.

Address: 0x000C + *n**0x4 (*n*=0 to 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FMSZ					CPOL	CPHA	PE	PP	FMSZ5	0	0	0	0	0	0
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 875. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)

Table 964. DSPI_CTAR*n*_SLAVE field descriptions

Field	Description
0:4 FMSZ	Frame Size The number of bits transferred per frame is equal to the concatenated field {CTAR*n*_SLAVE[FMSZ5], CTAR*n*_SLAVE[FMSZ]} value plus 1. The minimum valid FMSZ field value is 3 (when FMSZ6 is 0).
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase or TSB mode. (Master and Slave mode) Selects which edge of SCK causes data to change and which edge causes data to be captured. Devices must have identical clock phase settings for successful communication between serial devices. In Continuous SCK or TSB modes, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9 FMSZ5	MSB of Frame Size when DSI is used in 64-bit Mode The number of bits transferred per frame is equal to the concatenated field {CTAR*n*_SLAVE[FMSZ5], CTAR*n*_SLAVE[FMSZ]} value plus 1.

49.3.6 DSPI Status Register (DSPI_SR)

The status register contains DSPI status bits and interrupt/DMA request event flag bits.

Software can clear most of the flag bits in the SR by writing a ‘1’ to them (w1c). Writing a 0 to a flag bit has no effect. This register may not be writable in module disable mode due to the use of power saving mechanisms.

Address: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	SPITCF	EOQF	TFUF	DSITCF	TFFF	BSYF	CMDTCF	DPEF	SPEF	DDIF	RFOF	TFIWF	RFDF	CMDFFF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 876. DSPI Status Register (DSPI_SR)

Table 965. DSPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag. Indicates that all bits in a frame have been shifted out. 0 Transfer not complete. 1 Transfer complete.
1 TXRXS	TX and RX Status. Reflects the run status of the DSPI. 0 Transmit and receive operations are disabled (DSPI is in stopped state). 1 Transmit and receive operations are enabled (DSPI is in running state).
2 SPITCF	SPI Frame Transfer Complete Flag. Similar to TCF but only asserted on SPI frame transmission completion in CSI Mode This includes ITSB and ‘TSB in CSI’ modes, but not SPI or DSI Modes. 0 SPI frame transfer is not complete. 1 SPI frame transfer is complete.
3 EOQF	End of Queue Flag. Indicates that the last entry in a queue has been transmitted when the DSPI is in master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.

Table 965. DSPI_SR field descriptions (continued)

Field	Description
4 TFUF	Transmit FIFO Underflow Flag. Indicates an underflow condition in the TX FIFO. Transmit underflow is detected only for DSPI blocks in slave mode and SPI configuration. TFUF is set when the TX FIFO of a DSPI operating in SPI slave mode is empty and an external SPI master initiates a transfer. 0 No Tx FIFO underflow. 1 Tx FIFO underflow has occurred.
5 DSITCF	DSI Frame Transfer Complete Flag. Similar to TCF but only asserted on DSI frame transmission completion in CSI Mode. This includes ITSB and 'TSB in CSI' modes, but not SPI or DSI Modes. 0 DSI frame transfer is not complete. 1 DSI frame transfer is complete.
6 TFFF	Transmit FIFO Fill Flag. Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing '1' to it or by acknowledgement from the DMA controller to the TX FIFO full request. 0 Tx FIFO is full. 1 Tx FIFO is not full.
7 BSYF	Busy Flag. This bit is valid only when DSPI_MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmitting multiple data frames. This bit is not set for the last Data Frame of a Cyclic command Transfer or when DSPI_CTARE[DTCP] = 1. Refer to *Section 49.4.2.5: Command First In First Out (CMD FIFO) buffering mechanism*. This bit is valid only when DSPI_MCR[XSPI] is enabled. 0 No Cyclic Command Transfer in Progress. 1 Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for current cyclic command transfer.
8 CMDTCF	Command Transfer Complete Flag. Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when DSPI_CTARE[DTCP] = 1. CMDTCF remains set until it is cleared by writing a '1' to it. 0 Data Transfer by current Command not complete. 1 Data Transfer by current Command complete.
9 DPEF	DSI Parity Error Flag. Indicates that a DSI frame with parity error has been received. 0 No parity error. 1 Parity error has occurred.

Table 965. DSPI_SR field descriptions (continued)

Field	Description
10 SPEF	SPI Parity Error Flag. Indicates that a SPI frame with parity error has been received. 0 No parity error. 1 Parity error has occurred.
11 DDIF	DSI Data Received with Active Bits. Indicates that a DSI frame has been received with bits selected by DIMR with active polarity, defined by the DPIR. 0 No DSI data with active bits was received. 1 DSI data with active bits was received.
12 RFOF	Receive FIFO Overflow Flag. Indicates an overflow condition in the RX FIFO. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 TFIWF	Transmit FIFO Invalid Write Flag. Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. 0 No Invalid Data present in TX FIFO 1 Invalid Data present in TX FIFO since CMD FIFO is empty
14 RFDF	Receive FIFO Drain Flag. Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 Rx FIFO is empty. 1 Rx FIFO is not empty.
15 CMDFFF	Command FIFO Fill Flag. Provides a method for the DSPI to request more entries to be added to the CMD FIFO. The CMDFFF bit is set while the CMD FIFO is not full. The CMDFFF is cleared by writing a ‘1’ to it or by acknowledgement from the DMA controller to the CMD FIFO full request. 0 CMD FIFO is full. 1 CMD FIFO is not full.
16:19 TXCTR	TX FIFO Counter. Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the data part of PUSHR is written. The TXCTR is decremented every time the SPI data is transferred to the shift register.
20:23 TXNXTPTR	Transmit Next Pointer. Indicates which TX FIFO Entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.

Table 965. DSPI_SR field descriptions (continued)

Field	Description
24:27 RXCTR	RX FIFO Counter. Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28:31 POPNXTPTR	Pop Next Pointer. Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

49.3.7 DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

The DSPI_RSER controls DMA and interrupt requests.

Do not write to the RSER while the DSPI is in the Running state.

Address: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TCF_RE	CMDFFF_RE	SPITCF_RE	EOQF_RE	TFUF_RE	DSITCF_RE	TFFF_RE	TFFF_DIRS	CMDTCF_RE	DPEF_RE	SPEF_RE	DDIF_RE	RFOF_RE	TFIWF_RE	RFDF_RE	RFDF_DIRS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDFFF_DIRS	DDIF_DIRS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 877. DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

Table 966. DSPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable. Enables TCF flag in the Status register to generate an interrupt request. 0 Disabled. 1 Enabled.
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable. Enables the CMDFFF flag in the Status register to generate a request. DSPI_RSER[CMDFFF_DIRS] toggles either an interrupt or a DMA request. 0 Disabled. 1 Enabled.

Table 966. DSPI_RSER field descriptions (continued)

Field	Description
2 SPITCF_RE	SPI Frame Transmission Complete Request Enable. Enables SPITCF flag in the Status register to generate an interrupt request. 0 Disabled. 1 Enabled.
3 EOQF_RE	DSPI Finished Request Enable Enables the EOQF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
5 DSITCF_RE	DSI Frame Transmission Complete Request Enable. Enables DSITCF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the status register to generate a request. DSPI_RSER[TFFF_DIRS] toggles between either an interrupt or a DMA request. 0 Disabled. 1 Enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. DSPI_SR[TFFF] flag and DSPI_RSER[TFFF_RE] must be set. 0 Interrupt requests. 1 DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
9 DPEF_RE	DSI Parity Error Request Enable Enables the DPEF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
11 DDIF_RE	DSI data received with active bits Request Enable Enables the DDIF flag in the status register to generate an interrupt requests. 0 Disabled. 1 Enabled.

Table 966. DSPI_RSER field descriptions (continued)

Field	Description
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable Enables the TFIWF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the status register to generate a request. DSPIRSER[RFDF_DIRS] toggles either an interrupt or a DMA request. 0 Disabled 1 Enabled
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPI_SR[RFDF] flag and DSPI_RSER[RFDF_RE] must be set. 0 Interrupt request. 1 DMA request.
16 CMDFFF_DIRS	Command FIFO FIll DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPI_SR[CMDFFF] flag and DSPI_RSER[CMDFFF_RE] must be set. 0 Interrupt request. 1 DMA request.
17 DDIF_DIRS	DSI data received with active bits - DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPI_SR[DDIF] flag and DSPI_RSER[DDIF_RE] must be set. 0 Interrupt request. 1 DMA request.

49.3.8 DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)

PUSHR provides the means to write to the TX FIFO and CMD FIFO.

Data written to this register is transferred to:

- The TX FIFO for 8- or 16-bit writes to the Data field of PUSHR.
- The CMD FIFO for writes to the Command field of PUSHR.

In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO.

In slave mode, CMD FIFO is unused and the 16-bit Command field of PUSHR is reserved.

When Extended SPI Mode is not enabled (MCR[XSPI] = 0):

- TX FIFO and CMD FIFO must be filled simultaneously.
- Writes must be given to both Data and Command fields of PUSHR for every PUSHR operation.
- TX FIFO and CMD FIFO can be considered a single 32-bit FIFO.

When Extended SPI Mode is enabled (MCR[XSPI] = 1):

- TX FIFO and CMD FIFO can be written independently.
- A PUSHR Read Operation returns the topmost TX FIFO and CMD FIFO entries concatenated.

When DSPI Module is disabled, any writes to this register do not update the FIFO. Reads during Module disable mode return the last PUSHR write performed when Module was enabled.

Address: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CONT	CTAS			EOQ	CTCNT	PE_MASC	PP_MCSC	PCS							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 878. DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)

Table 967. DSPI_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable. (SPI master mode) Continuous Selection Format enables the selected PCS signals to remain asserted between transfers. PUSHR[EOQ] and PUSHR[CONT] bits cannot be asserted together in any Command Frame as EOQ signifies the last frame in a queue and the Last frame in a Continuous Transfer must have PUSHR[CONT] as 0. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1:3 CTAS	Clock and Transfer Attributes Select. Selects which CTAR (and corresponding CTARE register) to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI slave mode, CTAR0 is used. Refer to the Device Configuration chapter to determine how many CTARs this device has. Do not program a value in this field for a register that is not present. 000 CTAR0/CTARE0 001 CTAR1/CTARE1 010 CTAR2/CTARE2 011 CTAR3/CTARE3 100 CTAR4/CTARE4 101 CTAR5/CTARE5 110 CTAR6/CTARE6 111 CTAR7/CTARE7

Table 967. DSPI_PUSHR field descriptions (continued)

Field	Description
4 EOQ	End Of Queue Host software uses this bit to signal to the DSPI that the current SPI transfer is the last in a queue. DSPI_SR[EOQF] is set at the end of the transfer. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter. Clears the SPI_TCNT field in the Transfer Count register. The SPI_TCNT field is cleared before the DSPI starts transmitting the current SPI frame. Hence the current SPI frame causes this counter to increment by 1. 0 Do not clear the TCR[SPI_TCNT] field. 1 Clear the TCR[SPI_TCNT] field.
6 PE_MASC	Parity Enable or Mask t_{ASC} delay in the current frame PE: This bit enables parity bit transmission and parity reception check for the SPI frame. MASC: The current frame has the “after SCK” delay masked if this bit is asserted. Refer to *Section 49.4.6.6: Fast Continuous Selection Format* for more details. This bit is used as Mask t_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PE: No parity bit included/checked. MASC: t_{ASC} delay is not masked and the current frame has the after SCK delay. 1 PE: Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC: t_{ASC} delay is masked in the current frame.
7 PP_MCSC	Parity Polarity or Mask t_{CSC} delay in the next frame PP: controls the polarity of the parity bit transmitted and checked. MCSC: The next frame has the “PCS to SCK” delay masked if this bit is asserted. Refer to *Section 49.4.6.6: Fast Continuous Selection Format* for more details. This bit is used as Mask t_{CSC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PP: Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. MCSC: t_{CSC} delay is not masked and the next frame has the PCS to SCK delay. 1 PP: Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC: t_{CSC} delay is masked in the next frame.
8:15 PCS	Select which PCS signals are to be asserted for the transfer. Refer to the chip configuration chapter for the number of PCS signals used in this MCU. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16:31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command

49.3.9 DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

PUSHR provides the means to write to the TX FIFO.

Data written to this register is transferred to:

- The TX FIFO for 8- or 16-bit writes to the Data field of PUSHR.

In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO.

In slave mode, CMD FIFO is unused and the 16-bit Command field of PUSHR is reserved.

In Slave mode, up to 32-bit SPI frames may be queued for transmission/reception by enabling MCR[XSPI] mode.

Address: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 879. DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

Table 968. DSPI_PUSHR_SLAVE field descriptions

Field	Description
16:31 TXDATA	Transmit Data Holds SPI data to be transferred.

49.3.10 DSPI POP FIFO Register (DSPI_POPR)

POPR is used to read the RX FIFO.

8- or 16-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register generates a Transfer Error.

Address: 0x0038 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 880. DSPI POP FIFO Register (DSPI_POPR)

Table 969. DSPI_POPR field descriptions

Field	Description
0:31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

49.3.11 DSPI Transmit FIFO Registers (DSPI_TXFR*n*)

TXFR*n* provide visibility into the TX FIFO for debugging purposes.

Each read-only register is an entry in the TX FIFO. Reading the TXFR*n* registers does not alter the state of the TX FIFO.

Reading DSPI_TXFR*n* registers is invalid for a DSPI Master when used in Extended SPI (XSPI) mode.

Address: 0x003C + *n**0x4 (*n* = 0 to 3) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXCMD_TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 881. DSPI Transmit FIFO Registers (DSPI_TXFR*n*)

Table 970. DSPI_TXFR*n* field descriptions

Field	Description
0:15 TXCMD_TXDATA	Transmit Command or Transmit Data In master mode, the TXCMD field sets the transfer attributes for the SPI data. In slave mode, the TXDATA contains 16 MSB bits of the SPI data to be shifted out.
16:31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

49.3.12 DSPI Receive FIFO Registers (DSPI_RXFR*n*)

RXFR*n* provide visibility into the RX FIFO for debugging purposes.

Each read-only register is an entry in the RX FIFO. Reading the RXFR*n* registers does not alter the state of the RX FIFO.

MCR[MDIS] must be 0 when RXFR is read.

Address: 0x007C + *n**0x4 (*n* = 0 to 3) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 882. DSPI Receive FIFO Registers (DSPI_RXFR*n*)

Table 971. DSPI_RXFR*n* field descriptions

Field	Description
0:31 RXDATA	Receive Data Contains the received SPI data.

49.3.13 DSPI DSI Configuration Register 0 (DSPI_DSICR0)

DSICR0 selects various attributes associated with DSI and CSI Configurations.

Do not write to the DSICR0 while the DSPI is in the Running state.

Address: 0x00BC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FMSZ4	0	0	0	0	0	0	FMSZ5	0	ITSB	TSBC	TXSS	TPOL(1)	0	CID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DCONT	DSICTAS			DMS	PES	PE	PP	DPCSx							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 883. DSPI DSI Configuration Register 0 (DSPI_DSICR0)

1. This bit is reserved if the device does not have a hardware trigger input signal (HT).

Table 972. DSPI_DSICR0 field descriptions

Field	Description
1 FMSZ4	MSB of the frame size in master mode when DSI is used in 32-bit mode. If the bit is set, 16 is added to the frame size, as defined by the CTAR*n*[FMSZ] field. The CTAR*n* register is selected by the DSICTAS field.
8 FMSZ5	MSB of the frame size in master mode when DSI is used in 64-bit mode. If the bit is set, 32 is added to the frame size, as defined by the concatenation of {DSICR0[FMSZ4], CTAR*n*[FMSZ]} fields. The CTAR*n* register is selected by the DSICTAS field. Hence the final concatenated frame size using this bit is {DSICR0[FMSZ5], DSICR0[FMSZ4], CTAR*n*[FMSZ]}. This field is only valid when DSICR1[DSI64] is enabled.
10 ITSB	Interleaved TSB mode. Enables the Interleaved TSB mode of operation. When enabled, there is no priority among frames from the SPI/DSI. DSI frames are sent when either the previous transmission was an SPI frame or the TX_FIFO is empty. Frames are transmitted on every trigger whose source is selected by the TRG bit setting. ITSB mode requires DSICR0[TSBC] set and should be written only when MCR[HALT] is asserted. If ITSB bit is set without setting DSICR0[TSBC], DSPI operates in Normal Mode depending on MCR[DCONF] bit. Refer to *Section 49.4.10: Interleaved TSB (ITSB) Mode* for more details. 0 Disabled. 1 Enabled.
11 TSBC	Timed Serial Bus Configuration. Enables the Timed Serial Bus Configuration, which allows up to 64-bit data to be used. It also allows t_{DT} to be programmable. 0 Disabled. 1 Enabled.
12 TXSS	Transmit Data Source Select. Selects the source of data to be serialized. The source can be either: a) data from host software written to the DSPI DSI Alternate Serialization Data Register (ASDR1/0), or b) parallel input pin states latched into the DSPI DSI Serialization Data Register (SDR1/0). 0 SDR source. 1 ASDR source.
13 TPOL	Trigger Polarity Selects the active edge of the hardware trigger input signal (HT), initiating DSI frames transfer. 0 Falling edge initiates a transfer. 1 Rising edge initiates a transfer.
15 CID	Change In Data Transfer Enable When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out. Do not enable CID when DCONT is enabled. 0 Disabled. 1 Enabled.

Table 972. DSPI_DSICR0 field descriptions (continued)

Field	Description
16 DCONT	DSI Continuous Peripheral Chip Select Enable (DSI Master mode only) Enables the PCS signals to remain asserted between transfers. When the TSBC bit is set, DCONT has no effect. Do not enable DCONT when CID is enabled. 0 PCS signals return to inactive. 1 PCS signals remain asserted.
17:19 DSICTAS	DSI Clock and Transfer Attributes Select (DSI Master mode only) Selects the CTAR which provides transfer attributes for DSI frames. In DSI slave mode, CTAR1 is always selected.
20 DMS	Data Match Stop Stops DSI frame transmissions if DDIF flag is set in the status register. 0 Disabled. 1 Enabled.
21 PES	Parity Error Stop Stops DSI operation if a parity error occurred in the received DSI frame. 0 Disabled. 1 Enabled.
22 PE	Parity Enable Enables parity bit transmission and parity reception check for the DSI frames. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
23 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of ‘1’ bits in the transmitted frame is even. The SR[DPEF] bit is set if in the received frame number of ‘1’ bits is odd. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[DPEF] bit is set if in the received frame number of 1 bits is even.
24:31 DPCSx	DSI Peripheral Chip Select 0–7 Selects which of the PCS signals to assert during a DSI master mode transfer. 0 Negate PCS[x]. 1 Assert PCS[x].

49.3.14 DSPI DSI Serialization Data Register 0 (DSPI_SDR0)

Read-only SDR0 contains the states of the 32 LSB parallel input signals.

The states of these signals are latched into the SDR0 on the rising edge of every protocol clock.

When the TXSS bit in the DSICR0 is cleared, the data in the SDR0 is used as the source of the DSI frames.

When DSICR1[DSI64E] is set, SDR1 is the MSB half and SDR0 is the LSB half of the 64-bit SDR.

Address: 0x00C0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 884. DSPI DSI Serialization Data Register 0 (DSPI_SDR0)

Table 973. DSPI_SDR0 field descriptions

Field	Description
0:31 SER_DATA	Serialized Data Contains the signal states of the 32 LSB parallel input signals.

49.3.15 DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

ASDR0 is used by host software to write the 32 LSB of the data to be serialized.

When the TXSS bit in the DSICR0 is set, the data in the ASDR0 is the source of the DSI frames.

Writes to the ASDR0 take effect on the next frame boundary.

When DSICR1[DSI64E] is set, ASDR1 is the MSB half and ASDR0 is the LSB half of the 64-bit ASDR.

Address: 0x00C4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ASER_DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ASER_DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 885. DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

Table 974. DSPI_ASDR0 field descriptions

Field	Description
0:31 ASER_DATA	Alternate Serialized Data Holds the alternate 32 LSB of data to be serialized.

49.3.16 DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)

Read-only COMPR0 holds a copy of the 32 LSB of last transmitted DSI data.

DSI data is transferred to this register as it is loaded into the TX Shift Register.

When DSICR1[DSI64E] is set, COMPR1 is the MSB half and COMPR0 is the LSB half of the 64-bit COMPR.

Address: 0x00C8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 886. DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)

Table 975. DSPI_COMPR0 field descriptions

Field	Description
0:31 COMP_DATA	Compare Data Holds the 32 LSB of last serialized DSI data.

49.3.17 DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)

Read-only DDR0 holds the 32 LSB signal states for the parallel output signals.

Host software can read data from incoming DSI frames.

When DSICR1[DSI64E] is set, DDR1 is the MSB half and DDR0 is the LSB half of the 64-bit DDR.

Address: 0x00CC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 887. DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)

Table 976. DSPI_DDR0 field descriptions

Field	Description
0:31 DESER_DATA	Deserialized Data Holds 32 LSB of deserialized data presented as signal states to the parallel output signals.

49.3.18 DSPI DSI Configuration Register 1 (DSPI_DSICR1)

DSICR1 selects various attributes associated with TSB (and ITSB) Configuration.

Do not write to the DSICR1 while the DSPI is in the Running state.

Address: 0x00D0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	TSBCNT[5:0]						0	0	0	0	0	DSI64E	DSE1	DSE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRGPRD								DPCS1_x							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 888. DSPI DSI Configuration Register 1 (DSPI_DSICR1)

Table 977. DSPI_DSICR1 field descriptions

Field	Description
2:7 TSBCNT[5:0]	Timed Serial Bus Operation Count TSBCNT defines the length of the data frame When TSBC is set. The TSBCNT field valid value is from 3 to 63. The TSBCNT field selects the number of data bits to be shifted out during a transfer in TSB mode. Number of data bits in data frame = TSBCNT + 1.
13 DSI64E	DSI 64-bit Mode Enable Allows serialization and deserialization of Data frames up to 64 bits in size. The extended registers DSPI_SDR1, DSPI_ASDR1, DSPI_COMPR1, DSPI_DDR1, DSPI_SSR1, DSPI_PISR4-7, DSPI_DIMR1, DSPI_DPIR1 are valid in this mode. 0 Disabled. DSI Mode operates in the default 32-bit configuration 1 Enabled. DSI Mode operates in 64-bit configuration

Table 977. DSPI_DSICR1 field descriptions (continued)

Field	Description
14 DSE1	Data Select Enable 1 When DSICR0[TBSC] bit is set, the DSE1 bit enables insertion of the zero bit (Data Select) in the middle of the data frame. The insertion bit position is defined by the FMSZ field of CTAR*n* register, selected by the DSICR0[DSICTAS] field. If DSICR1[DSI64] is enabled, then the insertion bit position is defined by concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields. The TSBCNT field value must be greater than the FMSZ field value plus one for proper operation of the DSE1 bit. 0 No Zero bit is inserted in the middle of the data frame. 1 Zero bit is inserted at the middle of the data frame. Total number of bits in the data frame is increased by 1.
15 DSE0	Data Select Enable 0 When DSICR0[TSBC] bit is set, the DSE0 bit controls insertion of the zero bit (Data Select) in the beginning of the Data frame. 0 No Zero bit is inserted in the beginning of the frame. 1 Zero bit is inserted at the beginning of the data frame. Total number of bits in the data frame is increased by 1.
16:23 TRGPRD	Internal Trigger Period for the ITSB mode. When DSICR0[TRG] bit is cleared and ITSB Mode is enabled, this field determines the trigger period for the internal trigger in number of baud rate clock cycles. The trigger period should be calculated as: Trigger Period = 64 bits (DSI frame maximum size) + 2 selection bits (maximum) + t_{PF} (passive frame time) When DSICR1[DSI64E] is disabled, DSI frame maximum size is 32. Refer CTAR[FMSZ] field for minimum frame size allowed. Refer to Interleaved TSB (ITSB) Mode for more details. 0 – 4 These values are invalid since minimum Command frame size is 5. (that is 1 selection bit + minimum frame size which is 4) 5 – 255 Number of baud rate clock cycles to be used as the trigger period. The trigger period implemented is equal to TRGPRD + 1.
24:31 DPCS1_x	DSI Peripheral Chip Select 0–7 These bits define the PCSs to assert for the second part of the DSI frame when operating in TSB (and ITSB) configuration with dual receiver. The DPCS1 bits select which of the PCS signals to assert during the second part of the DSI frame. The DPCS1 bits control the assertions of the PCS signals only in TSB and ITSB mode. 0 Negate PCS[x]. 1 Assert PCS[x].

49.3.19 DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)

SSR0 is used to create a combined frame for transmission that contains bits from ASDR0 and from SDR0.

Each bit in the SSR0 register selects a corresponding bit to be serialized.

When DSICR0[TXSS] is set, the SSR0 register value has no effect.

When DSICR1[DSI64E] is set, SSR1 is the MSB half and SSR0 is the LSB half of the 64-bit SSR.

Address: 0x00D4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 889. DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)

Table 978. DSPI_SSR0 field descriptions

Field	Description
0:31 SS	Source Select Select the serialization source for the 32 LSB of DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. 0 The bit in the transmitted frame is taken from the parallel input pin. 1 The bit in the transmitted frame is taken from the ASDR0 register

49.3.20 DSPI DSI Parallel Input Select Register 0 (DSPI_PISR0)

PISR0–3 are used to select each data bit for the transmitted frame from 16 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31 (bit 0 is taken from parallel input 63 if DSICR1[DSI64E] is enabled). When the IPS0 is equal to +1, then bit 0 in the frame is taken from parallel input 1 and so on.

Note that PISR0–3 only preselect the parallel input pins. The final selection to the transmitted frame is made with the DSPI_SSR1/0 register bits or the DSICR0[TXSS] bit.

When DSICR1[DSI64E] is enabled, PISR4-7 registers are used to handle the extra 32 bits (32 MSB pins of the 64 parallel input pins).

Address: 0x00D8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS7				IPS6				IPS5				IPS4			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS3				IPS2				IPS1				IPS0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 890. DSPI DSI Parallel Input Select Register 0 (DSPI_PISR0)

Table 979. DSPI_PISR0 field descriptions

Field	Description
0:3 IPS7	Input Pin Select 7 Selects the parallel input pin for transmitted frame bit 7.
4:7 IPS6	Input Pin Select 6 Selects the parallel input pin for transmitted frame bit 6.
8:11 IPS5	Input Pin Select 5 Selects the parallel input pin for transmitted frame bit 5.
12:15 IPS4	Input Pin Select 5 Selects the parallel input pin for transmitted frame bit 4.
16:19 IPS3	Input Pin Select 3 Selects the parallel input pin for transmitted frame bit 3.
20:23 IPS2	Input Pin Select 2 Selects the parallel input pin for transmitted frame bit 2.
24:27 IPS1	Input Pin Select 1 Selects the parallel input pin for transmitted frame bit 1.
28:31 IPS0	Input Pin Select 0 Selects the parallel input pin for transmitted frame bit 0.

49.3.21 DSPI DSI Parallel Input Select Register 1 (DSPI_PISR1)

Address: Base + 0x00DC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS15				IPS14				IPS13				IPS12			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS11				IPS10				IPS9				IPS8			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 891. DSPI DSI Parallel Input Select Register 1 (DSPI_PISR1)

Table 980. DSPI_PISR1 field descriptions

Field	Description
0:3 IPS15	Input Pin Select 15 Selects the parallel input pin for transmitted frame bit 15.
4:7 IPS14	Input Pin Select 14 Selects the parallel input pin for transmitted frame bit 14.
8:11 IPS13	Input Pin Select 13 Selects the parallel input pin for transmitted frame bit 13.
12:15 IPS12	Input Pin Select 12 Selects the parallel input pin for transmitted frame bit 12.
16:19 IPS11	Input Pin Select 11 Selects the parallel input pin for transmitted frame bit 11.
20:23 IPS10	Input Pin Select 10 Selects the parallel input pin for transmitted frame bit 10.
24:27 IPS9	Input Pin Select 9 Selects the parallel input pin for transmitted frame bit 9.
28:31 IPS8	Input Pin Select 8 Selects the parallel input pin for transmitted frame bit 8.

49.3.22 DSPI DSI Parallel Input Select Register 2 (DSPI_PISR2)

Address: 0x00E0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS23				IPS22				IPS21				IPS20			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS19				IPS18				IPS17				IPS16			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 892. DSPI DSI Parallel Input Select Register 2 (DSPI_PISR2)

Table 981. DSPI_PISR2 field descriptions

Field	Description
0:3 IPS23	Input Pin Select 23 Selects the parallel input pin for transmitted frame bit 23.
4:7 IPS22	Input Pin Select 22 Selects the parallel input pin for transmitted frame bit 22.
8:11 IPS21	Input Pin Select 21 Selects the parallel input pin for transmitted frame bit 21.
12:15 IPS20	Input Pin Select 20 Selects the parallel input pin for transmitted frame bit 20.
16:19 IPS19	Input Pin Select 19 Selects the parallel input pin for transmitted frame bit 19.
20:23 IPS18	Input Pin Select 18 Selects the parallel input pin for transmitted frame bit 18.
24:27 IPS17	Input Pin Select 17 Selects the parallel input pin for transmitted frame bit 17.
28:31 IPS16	Input Pin Select 16 Selects the parallel input pin for transmitted frame bit 16.

49.3.23 DSPI DSI Parallel Input Select Register 3 (DSPI_PISR3)

Address: 0x00E4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	IPS31				IPS30				IPS29				IPS28			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	IPS27				IPS26				IPS25				IPS24			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 893. DSPI DSI Parallel Input Select Register 3 (DSPI_PISR3)

Table 982. DSPI_PISR3 field descriptions

Field	Description
0:3 IPS31	Input Pin Select 31 Selects the parallel input pin for transmitted frame bit 31.
4:7 IPS30	Input Pin Select 30 Selects the parallel input pin for transmitted frame bit 30.
8:11 IPS29	Input Pin Select 29 Selects the parallel input pin for transmitted frame bit 29.
12:15 IPS28	Input Pin Select 28 Selects the parallel input pin for transmitted frame bit 28.
16:19 IPS27	Input Pin Select 27 Selects the parallel input pin for transmitted frame bit 27.
20:23 IPS26	Input Pin Select 26 Selects the parallel input pin for transmitted frame bit 26.
24:27 IPS25	Input Pin Select 25 Selects the parallel input pin for transmitted frame bit 25.
28:31 IPS24	Input Pin Select 24 Selects the parallel input pin for transmitted frame bit 24.

49.3.24 DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

DIMR0 selects bits in the 32 LSB of received DSI frame to be checked to generate the DDI interrupt.

When DSICR1[DSI64E] is set, DIMR1 is the MSB half and DIMR0 is the LSB half of the 64-bit DIMR.

Address: 0x00E8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 894. DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

Table 983. DSPI_DIMR0 field descriptions

Field	Description
0:31 MASK	Mask 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame produces a DDI interrupt if the data bit matches the configured polarity.

49.3.25 DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

DPIR0 defines which data bit value in the 32 LSB of received DSI frame generates the DDI interrupt.

When DSICR1[DSI64E] is set, DPIR1 is the MSB half and DPIR0 is the LSB half of the 64-bit DPIR.

Address: 0x00EC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 895. DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

Table 984. DSPI_DPIR0 field descriptions

Field	Description
0:31 DP	Data Polarity 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

49.3.26 DSPI DSI Serialization Data Register 1 (DSPI_SDR1)

Read-only SDR1 contains the states of the 32 MSB parallel input signals.

The states of these signals are latched into the SDR1 on the rising edge of every protocol clock.

When DSICR0[TXSS] is cleared, the data in SDR1 and SDR0 is the source of the DSI frames.

The concatenation of {DSPI_SDR1, DSPI_SDR0} provides the 64-bit data to be serialized.

This register is valid only when DSICR1[DSI64E] is set.

Address: 0x00F0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 896. DSPI DSI Serialization Data Register 1 (DSPI_SDR1)

Table 985. DSPI_SDR1 field descriptions

Field	Description
0:31 SER_DATA	Serialized Data

49.3.27 DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)

ASDR1 is used by host software to write the 32 MSB of the 64-bit data to be serialized.

When DSICR0[TXSS] is set, the data in ASDR1 and ASDR0 is the source of the DSI frames.

Writes to the ASDR1 take effect on the next frame boundary.

The concatenation of {DSPI_ASDR1, DSPI_ASDR0} provides the 64-bit alternate serialization data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x00F4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ASER_DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ASER_DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 897. DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)

Table 986. DSPI_ASDR1 field descriptions

Field	Description
0:31 ASER_DATA	Alternate Serialized Data

49.3.28 DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)

Read-only COMPR1 holds a copy of the 32 MSB of the last transmitted 64-bit DSI data.

The upper 32 bits of the 64-bit DSI data is transferred to this register as it is loaded into the TX Shift register.

The concatenation of {DSPI_COMPR1, DSPI_COMPR0} provides the 64-bit transmit comparison data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x00F8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 898. DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)

Table 987. DSPI_COMPR1 field descriptions

Field	Description
0:31 COMP_DATA	Compare Data

49.3.29 DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)

Read-only DDR1 holds the 32 MSB signal states for the 64 parallel output signals.

Host software can read data from incoming DSI frames.

The concatenation of {DSPI_DDR1, DSPI_DDR0} provides the 64-bit Deserialized data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x00FC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 899. DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)

Table 988. DSPI_DDR1 field descriptions

Field	Description
0:31 DESER_DATA	Deserialized Data

49.3.30 DSPI DSI Serialization Source Select Register 1 (DSPI_SSR1)

SSR1 is used to create a combined frame for transmission that contains bits from ASDR1 and from SDR1.

Each bit in the SSR1 register selects a corresponding bit to be serialized.

When DSICR0[TXSS] is set, the SSR1 register value has no effect.

The concatenation of {DSPI_SSR1, DSPI_SSR0} provides the 64-bit Serialization Source Select value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x0100 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 900. DSPI DSI Serialization Source Select Register (DSPI_SSR)

Table 989. DSPI_SSR field descriptions

Field	Description
0:31 SS	Source Select 0 The bit in the transmitted frame is taken from the parallel input pin. 1 The bit in the transmitted frame is taken from the ASDR1 register

49.3.31 DSPI DSI Parallel Input Select Register 4 (DSPI_PISR4)

PISR4–7 are used to select each data bit for the transmitted frame from 16 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31 (bit 0 is taken from parallel input 63 when DSICR1[DSI64E] is enabled). When the IPS0 is equal to +1, then bit 0 in the frame is taken from parallel input 1 and so on.

Note that PISR0–7 only preselect the parallel input pins. The final selection to the transmitted frame is made with the SSR0 and SSR1 register bits or the DSICR0[TXSS] bit.

When 64-bit DSI mode is enabled, PISR4-7 registers are used to handle the extra 32 bits (32 MSB pins of the 64 parallel input pins). If DSICR1[DSI64E] is disabled, then DSPI_PISR4-7 registers are invalid.

Address: 0x0104 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS39				IPS38				IPS37				IPS36			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS35				IPS34				IPS33				IPS32			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 901. DSPI DSI Parallel Input Select Register 4 (DSPI_PISR4)

Table 990. DSPI_PISR4 field descriptions

Field	Description
0:3 IPS39	Input Pin Select 39 Selects the parallel input pin for transmitted frame bit 39.
4:7 IPS38	Input Pin Select 38 Selects the parallel input pin for transmitted frame bit 38.
8:11 IPS37	Input Pin Select 37 Selects the parallel input pin for transmitted frame bit 37.
12:15 IPS36	Input Pin Select 36 Selects the parallel input pin for transmitted frame bit 36.
16:19 IPS35	Input Pin Select 35 Selects the parallel input pin for transmitted frame bit 35.
20:23 IPS34	Input Pin Select 34 Selects the parallel input pin for transmitted frame bit 34.
24:27 IPS33	Input Pin Select 33 Selects the parallel input pin for transmitted frame bit 33.
28:31 IPS32	Input Pin Select 32 Selects the parallel input pin for transmitted frame bit 32.

49.3.32 DSPI DSI Parallel Input Select Register 5 (DSPI_PISR5)

Address: 0x0108 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	IPS47				IPS46				IPS45				IPS44			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	IPS43				IPS42				IPS41				IPS40			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 902. DSPI DSI Parallel Input Select Register 5 (DSPI_PISR5)

Table 991. DSPI_PISR5 field descriptions

Field	Description
0:3 IPS47	Input Pin Select 47 Selects the parallel input pin for transmitted frame bit 47.
4:7 IPS46	Input Pin Select 46 Selects the parallel input pin for transmitted frame bit 46.
8:11 IPS45	Input Pin Select 45 Selects the parallel input pin for transmitted frame bit 45.
12:15 IPS44	Input Pin Select 44 Selects the parallel input pin for transmitted frame bit 44.
16:19 IPS43	Input Pin Select 43 Selects the parallel input pin for transmitted frame bit 43.
20:23 IPS42	Input Pin Select 42 Selects the parallel input pin for transmitted frame bit 42.
24:27 IPS41	Input Pin Select 41 Selects the parallel input pin for transmitted frame bit 41.
28:31 IPS40	Input Pin Select 40 Selects the parallel input pin for transmitted frame bit 40.

49.3.33 DSPI DSI Parallel Input Select Register 6 (DSPI_PISR6)

Address: 0x010C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS55				IPS54				IPS53				IPS52			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS51				IPS50				IPS49				IPS48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 903. DSPI DSI Parallel Input Select Register 6 (DSPI_PISR6)

Table 992. DSPI_PISR6 field descriptions

Field	Description
0:3 IPS55	Input Pin Select 55 Selects the parallel input pin for transmitted frame bit 55.
4:7 IPS54	Input Pin Select 54 Selects the parallel input pin for transmitted frame bit 54.
8:11 IPS53	Input Pin Select 53 Selects the parallel input pin for transmitted frame bit 53.
12:15 IPS52	Input Pin Select 52 Selects the parallel input pin for transmitted frame bit 52.
16:19 IPS51	Input Pin Select 51 Selects the parallel input pin for transmitted frame bit 51.
20:23 IPS50	Input Pin Select 50 Selects the parallel input pin for transmitted frame bit 50.
24:27 IPS49	Input Pin Select 49 Selects the parallel input pin for transmitted frame bit 49.
28:31 IPS48	Input Pin Select 48 Selects the parallel input pin for transmitted frame bit 48.

49.3.34 DSPI DSI Parallel Input Select Register 7 (DSPI_PISR7)

Address: 0x0110 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	IPS63				IPS62				IPS61				IPS60			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	IPS59				IPS58				IPS57				IPS56			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 904. DSPI DSI Parallel Input Select Register 7 (DSPI_PISR7)

Table 993. DSPI_PISR7 field descriptions

Field	Description
0:3 IPS63	Input Pin Select 63 Selects the parallel input pin for transmitted frame bit 63
4:7 IPS62	Input Pin Select 62 Selects the parallel input pin for transmitted frame bit 62
8:11 IPS61	Input Pin Select 61 Selects the parallel input pin for transmitted frame bit 61
12:15 IPS60	Input Pin Select 60 Selects the parallel input pin for transmitted frame bit 60
16:19 IPS59	Input Pin Select 59 Selects the parallel input pin for transmitted frame bit 59
20:23 IPS58	Input Pin Select 58 Selects the parallel input pin for transmitted frame bit 58
24:27 IPS57	Input Pin Select 57 Selects the parallel input pin for transmitted frame bit 57
28:31 IPS56	Input Pin Select 56 Selects the parallel input pin for transmitted frame bit 56

49.3.35 DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)

DIMR1 selects bits from the 32 MSB in the received 64-bit DSI frame to be checked to generate the DDI interrupt.

The concatenation of {DSPI_DIMR1, DSPI_DIMR0} provides the 64-bit Deserialized Data Interrupt Mask value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x0114 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 905. DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)

Table 994. DSPI_DIMR1 field descriptions

Field	Description
0:31 MASK	Mask 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.

49.3.36 DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

DPIR1 defines which of the 32 MSB value in the received 64-bit DSI frame generates the DDI interrupt.

The concatenation of {DSPI_DPIR1, DSPI_DPIR0} provides the 64-bit Deserialized Data Polarity value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x0118 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 906. DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

Table 995. DSPI_DPIR1 field descriptions

Field	Description
0:31 DP	Data Polarity 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

49.3.37 DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

CTARE registers are used to define the extended transfer attributes for an SPI frame.

These registers are valid only when DSPI_MCR[XSPI] is set.

When the DSPI is configured as:

- an SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used.
- an SPI bus slave, the CTARE0 register is used.

Address: 0x011C + *n**0x4 (*n* = 0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FMSZE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DTCP										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 907. DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

Table 996. DSPI_CTARE*n* field descriptions

Field	Description
15 FMSZE	Frame Size Extended This field concatenated with DSPI_CTAR[FMSZ] defines the frame size of the SPI frames to be transmitted. Frame size is the concatenation of {DSPI_CTAR[FMSZ], DSPI_CTARE[FMSZE]} + 1. 0 Default Mode. Up to 16-bit SPI frames can be transferred. 1 Up to 32-bit SPI frames can be transferred. Each frame transfer is a result of 2 TX FIFO Pops.
21:31 DTCP	Data Transfer Count Preload This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame that selected this DSPI_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

49.3.38 DSPI Status Register Extended (DSPI_SREX)

The DSPI_SREX contains status fields that reflect the DSPI status and indicate the occurrence of events.

This register is not writable.

Address: 0x013C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CMDCTR				CMDNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 908. DSPI Status Register Extended (DSPI_SREX)

Table 997. DSPI_SREX field descriptions

Field	Description
24:27 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
28:31 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

- DSPI DSI Configuration Register 0 (DSPI_DSICR0)
- DSPI DSI Configuration Register 1 (DSPI_DSICR1)
- DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)
- DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)
- DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)
- DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

49.4 Functional description

The Deserial Serial Peripheral Interface (DSPI) block supports full-duplex, synchronous serial communications between MCUs and peripheral devices. The DSPI can also be used to reduce the number of pins required for I/O by serializing and deserializing up to 64 parallel input/output signals. All communications are done with SPI-like protocol.

The DSPI has the following configurations:

- SPI Configuration in which the DSPI operates as a basic SPI or a queued SPI.
- DSI Configuration in which the DSPI serializes and deserializes parallel input/output signals or bits from memory-mapped register.
- CSI Configuration in which the DSPI combines the functionality of the SPI and DSI configurations.

The DCONF field in the DSPI Module Configuration Register (MCR) determines the DSPI Configuration. Refer to *Section 49.3.1: DSPI Module Configuration Register (DSPI_MCR)* for the DSPI configuration values.

The CTAR*n* registers hold clock and transfer attributes. The SPI configuration allows CTAR selection on a frame by frame basis by setting a field in the SPI command. Extended SPI Mode (DSPI_MCR[XSPI]) further allows the usage of CTARE*n* (CTAR*n* Extended) registers to send multiple Data frames (of up to 32-bit frame size) using a single Command frame.

DSI configuration statically selects which CTAR to use. In CSI, Configuration priority logic determines if SPI data or DSI data is transferred and dictates which CTAR is used for the data transfer.

Refer to *Section 49.3.4: DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTARn)* for information on CTARs fields.

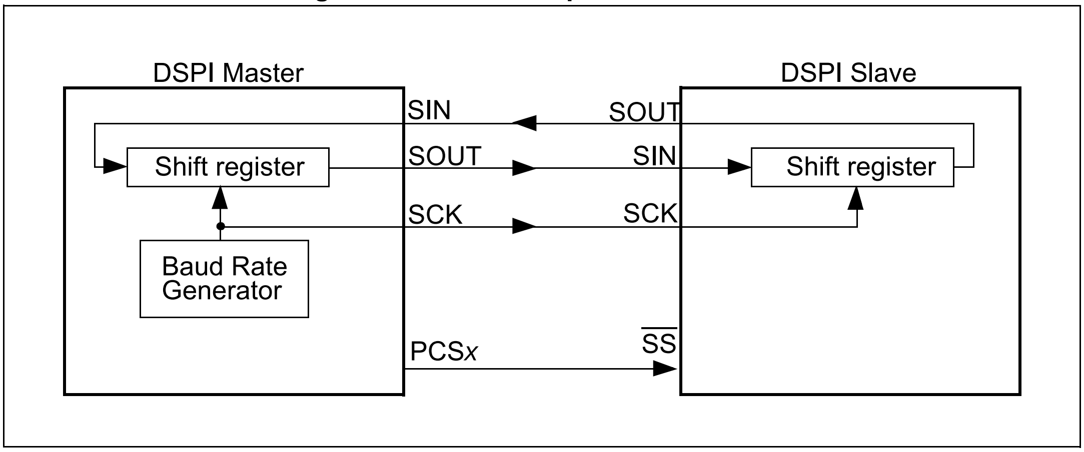
Refer to *Section 49.3.37: DSPI Clock and Transfer Attributes Register Extended (DSPI_CTAREn)* for information on CTAREs fields.

Typical master to slave connections are shown in *Figure 909*.

When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave: the data that was in the master shift register is now in the shift register of the slave, and vice versa.

At the end of a transfer, the TCF bit in the Status register is set to indicate a completed transfer.

Figure 909. SPI serial protocol overview



Generally, more than one slave device can be connected to the DSPI master.

Eight Peripheral Chip Select (PCS) signals of the DSPI masters can be used to select which of the slaves to communicate with. Refer to the chip configuration chapter for the number of PCS signals used in this MCU.

The three DSPI configurations share transfer protocol and timing properties which are described independently of the configuration in *Section 49.4.6: Transfer formats*.

The transfer rate and delay settings are described in *Section 49.4.5: DSPI baud rate and clock delay generation*.

49.4.1 Start and Stop of DSPI transfers

The DSPI has two operating states that are independent of DSPI configuration:

- STOPPED is the default DSPI state: no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results.
- RUNNING is the state when serial transfers take place.

DSPI_SR[TXRXS] indicates what state the DSPI in (‘1’ = RUNNING state).

The DSPI is started (DSPI transitions to RUNNING) when all of the following conditions are true:

- SR[EOQF] bit is clear
- MCU is not in the debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The DSPI stops (transitions from RUNNING to STOPPED) after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- MCU in the debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

49.4.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The SPI frames can be 32 bits long.

The DSPI is in SPI Configuration when the DCONF field in the MCR is 0b00.

The host CPU or a DMA controller transfers the SPI data from memory external to DSPI RAM queues to a transmit FIFO (TX FIFO) buffer.

Received data is stored in entries in the Receive FIFO (RX FIFO) buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the DSPI.

The FIFO buffers operation is described in:

- *Section 49.4.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*,
- *Section 49.4.2.5: Command First In First Out (CMD FIFO) buffering mechanism*,
- *Section 49.4.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism*.

The interrupt and DMA request conditions are described in *Section 49.4.12: Interrupts/DMA requests*.

The SPI Configuration supports two block-specific modes:

- Master mode: the DSPI initiates and controls the transfer according to the fields of the executing CMD FIFO entry.
- Slave mode: the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field space is reserved.

49.4.2.1 Master mode

In SPI Master mode, the DSPI initiates the serial transfers by controlling the Serial Communications Clock (SCK) and the Peripheral Chip Select (PCS) signals.

The executing SPI CMD FIFO entry determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See DSPI PUSH FIFO Register (PUSHR) for details on the SPI command fields.

The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin.

In SPI Master mode, each SPI frame to be transmitted has a command associated with it for transfer attribute control on a frame-by-frame basis.

In Extended SPI Master mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. Extended SPI Mode allows for larger frame sizes of up to 32 bits.

49.4.2.2 Slave mode

In SPI slave mode, the DSPI responds to transfers initiated by a SPI bus master. The DSPI does not initiate transfers.

Certain transfer attributes such as clock polarity, clock phase and frame size must be set in CTAR0 and CTARE0 for successful communication with an SPI master. The data is shifted out with MSB first; shifting out of LSB is not supported in this mode.

49.4.2.3 FIFO Disable operation

The FIFO Disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO.

The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled.

The Transmit and Receive side of the FIFOs are disabled separately:

- MCR[DIS_TXF] disables TX FIFO and CMD FIFO.
  - TFFF, TFUF, CMDFFF, CMDCTR and TXCTR fields in the status register and the extended status register behave as if there is a one-entry FIFO.
  - The contents of the TXFR registers and TXNXTPTR and CMDNXTPTR are undefined.
- MCR[DIS_RXF] bit disables the RX FIFO.
  - The RFDF, RFOF and RXCTR fields in the status register behave as if there is a one-entry FIFO.
  - The contents of the RXFR registers and POPNXTPTR are undefined.

The FIFO Disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

49.4.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission.

The TX FIFO holds 4 words, each consisting of SPI data that is added to the TX FIFO by writing to the Data field of DSPI PUSH FIFO Register (PUSHR).

The number of entries in the TX FIFO is device-specific.

TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the TX FIFO. TXCTR is updated every time an 8 or 16-bit write takes place to the Data field of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates which TX FIFO Entry will be transmitted during the next transfer. This field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

The TXFR*n* registers are invalid in Extended SPI Mode as TX FIFO and CMD FIFO are used separately.

49.4.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set.When TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete, the TFFF bit is cleared. Writing a ‘1’ to the TFFF bit also clears it. The DSPI ignores attempts to push data to a full TX FIFO; the state of the TX FIFO does not change and no error condition is indicated.

The TFFF can generate a DMA request or an interrupt request. Refer to *Section 49.4.12.2: Transmit FIFO Fill interrupt or DMA request* for details.

49.4.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one.

When Extended SPI Mode (DSPI_MCR[XSPI]) is enabled and the frame size of SPI Data to be transmitted is more than 16 bits, then two Data entries are popped from TX FIFO simultaneously and transferred to the shift register. The first of the two popped entries forms the 16 LSB bits of the SPI frame. Such an operation also causes TX FIFO Counter to decrement by two.

At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a ‘1’ to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a DSPI slave while the slave’s DSPI TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave’s SR is set. Refer to *Section 49.4.12.9: Transmit FIFO Underflow interrupt request* for details.

49.4.2.5 Command First In First Out (CMD FIFO) buffering mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission.

The CMD FIFO holds four entries, each representing command fields. The number of entries in the CMD FIFO is device-specific.

SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (DSPI_MCR[XSPI]) is disabled:

- The TX FIFO and CMD FIFO must be filled together as every CMDFIFO entry has a corresponding single TXFIFO entry attached to it.

When Extended SPI Mode (DSPI_MCR[XSPI]) is enabled:

- The TX FIFO and CMD FIFO can be filled independently as every CMDFIFO entry can have multiple TXFIFO entries attached to it.
- The CTARE[DTCP] field decides the number of SPI Data Frames of size {FMSZE, FMSZ} to be transmitted using the current Command Entry. FMSZ and FMSZE fields are given in the CTAR/CTARE registers respectively, pointed by the CTAS field in the Command frame.
- The amount of time a command entry is in use is known as a Command Cycle. The busy flag DSPI_SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO counter field (CMDCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the CMD FIFO. The TXCTR is updated every time an 8 or 16-bit write takes place on the lower half of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The CMDNXTPTR field indicates which CMD FIFO entry is used during the next command cycle. This field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

The TXFR*n* registers are invalid in Extended SPI Mode as TX FIFO and CMD FIFO are used separately.

49.4.2.6 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SIN pin.

The RX FIFO holds four received SPI data frames. The number of entries in the RX FIFO is device-specific.

SPI data is added to the RX FIFO on completion of a transfer when the received data in the shift register is transferred into the RX FIFO.

SPI data are removed (popped) from the RX FIFO by reading the DSPI POP FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the status register points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in number of 32-bit registers. For example, POPNXTPTR = 2 means that the RXFR2 contains the received SPI data that will be returned when POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR register number and it rolls over after reaching the maximum.

49.4.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register.

While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time a SPI frame is transferred to the RX FIFO the RX FIFO counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the status register is set to indicating an overflow condition. Consequently:

- If the DSPI_MCR[ROOE] bit is set, the incoming data is shifted in to the shift register.
- If the DSPI_MCR[ROOE] bit is cleared, the incoming data is ignored.

49.4.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the DSPI POP FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one.

Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data read from the empty RX FIFO is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the status register is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete, or by writing a ‘1’ to it.

49.4.3 Deserial Serial Interface (DSI) configuration

The DSI Configuration supports pin count reduction by serializing parallel input signals or register bits and shifting them out in a SPI-like protocol. The timing and transfer protocol is described in *Section 49.4.6: Transfer formats*. The received serial frames are converted to a parallel form (deserialized) and placed on the parallel output signals or in the DSPI_DDR. The various features of the DSI Configuration are set in DSPI DSI Configuration Registers (DSPI_DSICR1/0).

The DSI frames can be from 4 to 32 bits, or 64 bits when DSICR1[DSI64E] is enabled.

49.4.3.1 DSI master mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has four different conditions (described in *Section 49.4.3.5: DSI transfer initiation control*) that can initiate a transfer:

- Continuous.
- Change in data.
- Trigger signal.
- Trigger signal combined with a change in data.

Transfer attributes are set during initialization. DSICR0[DSICTAS] field determines which of the DSPI_CTARs will control the transfer attributes.

49.4.3.2 Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by a SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers.

Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI_CTAR1.

The data is shifted out with MSB first.

49.4.3.3 DSI serialization

The DSI configuration from 4 to 16 bits can be serialized using two different sources selected by DSICR0[TXSS]:

- DSPI DSI Serialization Data Register (DSPI_SDR1/0). The DSPI_SDR0 holds the latest 32 parallel input signal values which is sampled at every rising edge of the protocol clock.
  When DSICR1[DSI64E] is enabled, 64-bit DSI frames are supported and hence DSPI_SDR1 holds the 32 MSB parallel input signal values which is sampled at every rising edge of the protocol clock.
- DSPI DSI Alternate Serialization Data Register (DSPI_ASDR1/0). The DSPI_ASDR1/0 registers are written by host software and used as an alternate source of serialized data. DSPI_ASDR1 is only usable when DSICR1[DSI64E] is enabled.

The DSPI has to sample the (SDR/ASDR) inputs before it transmits a DSI frame. Due to the asynchronous clock structure within DSPI, the sampling of inputs come under Register Clock domain whereas DSI frame transmission logic lies in Protocol Clock domain. Hence the sampled data (from ASDR1/0 or SDR1/0) must be synchronized into the Protocol Domain before being loaded into the Shift register for transmission.

To prevent Clock Domain clocking issues, SDR/ASDR inputs should not change while they are being sampled into the Protocol Domain. This is assured in the following manner:

- For the ASDR0 and ASDR1 inputs:
  - The initial write performed on ASDR register causes the 32-bit ASDR input to be latched into a temporary register which is then locked (further latching is prevented) until this data gets synchronized into the Protocol Domain.
  - Once synchronized, the lock is removed and the next ASDR write triggers this process again. If a write is performed on ASDR while the lock is asserted, this is processed as soon as the previous synchronization process completes and the lock is de-asserted. If multiple writes are performed on ASDR when lock is asserted after the first write, only the last value is considered for synchronization once the lock is de-asserted.
- For the SDR0 and SDR1 inputs:
  - The process is the same as ASDR with the only difference being that SDR inputs are delayed internally by a single clock to detect a change in the SDR input pins.

Note: If SDR0 and SDR1 inputs are both changed at the same time instant, then the synchronization logic ensures that both SDR0 and SDR1 contribute to the same DSI frame.

Once the synchronization process for a particular Deserial input begins, it takes up to five Register clock periods and five Protocol clock periods before the synchronization process for the next Deserial input can begin. Hence the writes performed on ASDR0/1 and SDR0/1 are constrained with this timing.

The DSPI_PISR0–3 registers (DSPI_PISR0–7 registers if DSICR1[DSI64E] is enabled) allow changing of the relative position of the parallel input pins in the transmitted frame. Each transmitted frame bit can be selected from 16 adjacent parallel inputs by writing IPS*n* fields. The IPS*n* field is treated as a 4-bit integer number, representing numbers from -8 to 7. The parallel input pin number, selected by IPS*n* field is defined by the sum of the IPS*n* field number (*n*) and the IPS*n* field value. If the operation result is negative the number 32 should be added. If the result is higher than 32, 32 should be subtracted from the result. When DSICR1[DSI64E] is enabled, if the above operation result is negative, the number 64 should be added. If the result is higher than 64, 64 should be subtracted from the result.

For example, IPS0, set to minus 1 (binary 1111), preselects parallel input 31 to 0 position in the transmitted frame. (This preselects parallel input 63 to 0 position in the transmitted frame when DSICR1[DSI64E] is set).

IPS6, set to 3 (binary 0011), preselects parallel input 9 to be bit number 6 in the transmitted frame, while the value minus 2 (1110) preselects parallel input 4.

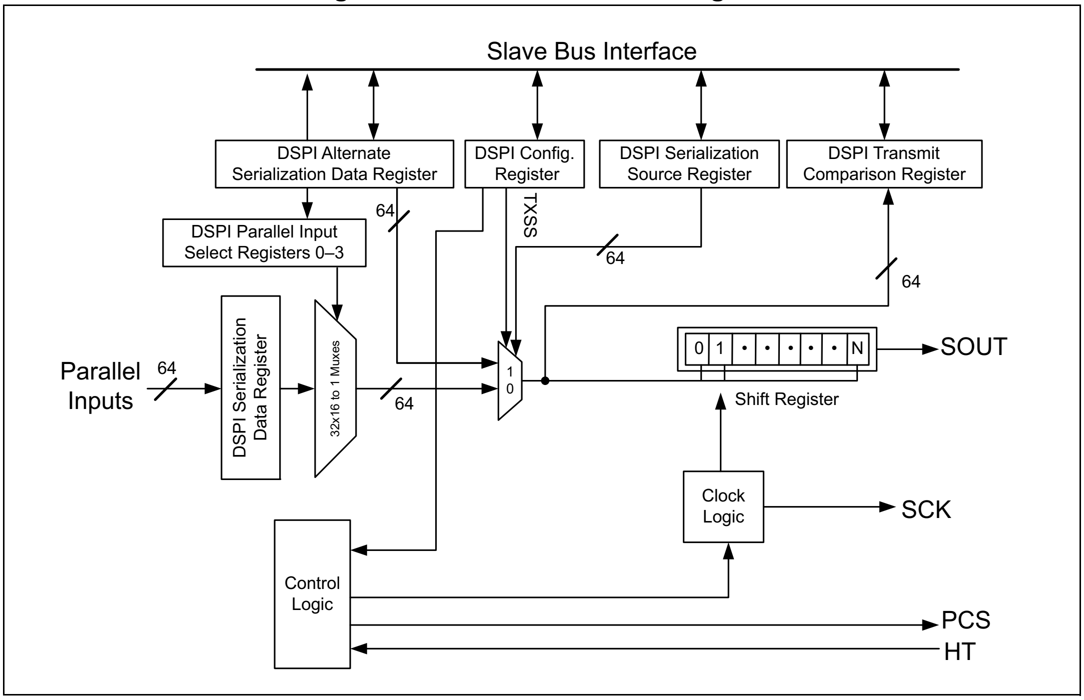
IPS31, set to minus 8 (binary 1000), preselects parallel input 23 to be bit number 31 in the transmitted frame.

The parallel input pin state to be transmitted should be selected by TXSS or DSPI_SSR1/0 registers and the frame size should be higher than the bit position in the preselected frame.

DSPI_SSR1/0 registers provide additional way of creating the frame for transmission. Each bit from this register is ORed with the TXSS bit and controls individual transmitted bit source. This way, the transmitted frame can have any combination of the DSPI_SDR1/0 and DSPI_ASDR1/0 bits. This feature allows control SPI based devices, requiring control and data fields in the frame. The control field may come from DSPI_ASDR1/0 registers set by the CPU, while data field can be generated by device peripheral modules, like PWM timers.

A copy of the last DSI frame shifted out of the Shift register is stored in the DSPI DSI Transmit Comparison Registers (DSPI_COMPR1/0). This register provides added visibility for debugging and it serves as a reference for transfer initiation control. *Figure 910* shows the DSI serialization logic.

Figure 910. DSI serialization diagram

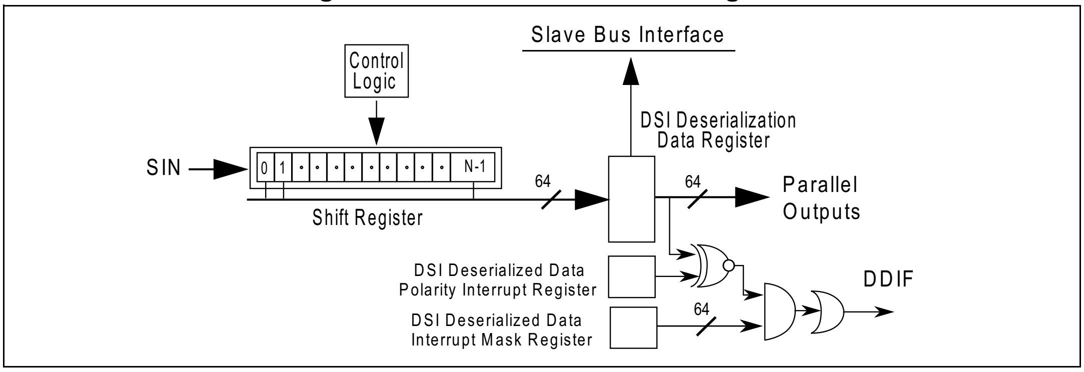


49.4.3.4 DSI deserialization

When all bits in a DSI frame have been shifted in, the frame is copied to the DSPI DSI Deserialization Data Register (DSPI_DDR1/0). This register presents the deserialized data as parallel output signal values. The DSPI_DDR1/0 is memory-mapped to allow host software to read the deserialized data directly.

The received data is bit-wise compared to the value of the DSI Deserialized Data Polarity Interrupt Register (DSPI_DPIR1/0), bit-wise ANDed with DSI Deserialized Interrupt Mask Register (DSPI_DIMR1/0) and the results ORed to produce DDIF flag in the DSPI_SR. Which in turn can cause DDIF interrupt request or DMA request based on RSER[DDIF_RE] and RSER[DDIF_DIRS] and/or stop DSI frame transmissions if DSICR0[DMS] bit is set. *Figure 911* shows the DSI Deserialization logic.

Figure 911. DSI deserialization diagram



49.4.3.5 DSI transfer initiation control

Data transfers for a master DSPI in DSI configuration are initiated by a condition. The transfer initiation conditions are selected by the TRRE and CID bits in the DSPI_DSICR0 register. Table 998 lists the four transfer initiation conditions.

Table 998. DSI data transfer initiation control

DSPI_DSICR0 bits		Transfer initiation control
TRRE	CID	
0	0	Continuous
0	1	Change in Data
1	0	Triggered
1	1	Triggered or Change in Data

49.4.3.5.1 Continuous control

For continuous control, a new DSI frame shifts out when the previous transfer cycle has completed and the delay after transfer (t_{DT}) has elapsed.

49.4.3.5.2 Change in data control

For change in data control, a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in the DSPI_COMPR1/0. When the data selected for the transfer from the DSPI_SDR1/0 and DSPI_ASDR1/0 registers is different from the data in the DSPI_COMPR1/0 a new DSI frame is transmitted.

49.4.3.5.3 Triggered control

For Triggered Control, initiation of a transfer is controlled by the Hardware Trigger signal (HT). DSICR0[TPOL] bit selects the active edge of HT. For HT to have any affect, DSICR0[TRRE] bit must be set.

49.4.3.5.4 Change in data or triggered control

For change in data or triggered control, initiation of a transfer is controlled by the detection of a change in data to be serialized or by the HT signal.

It is not advisable to configure a DSPI slave with the combination DSICR0[DCONT] = 1, DSICR0[CID] = 1; since the following statement then holds true:

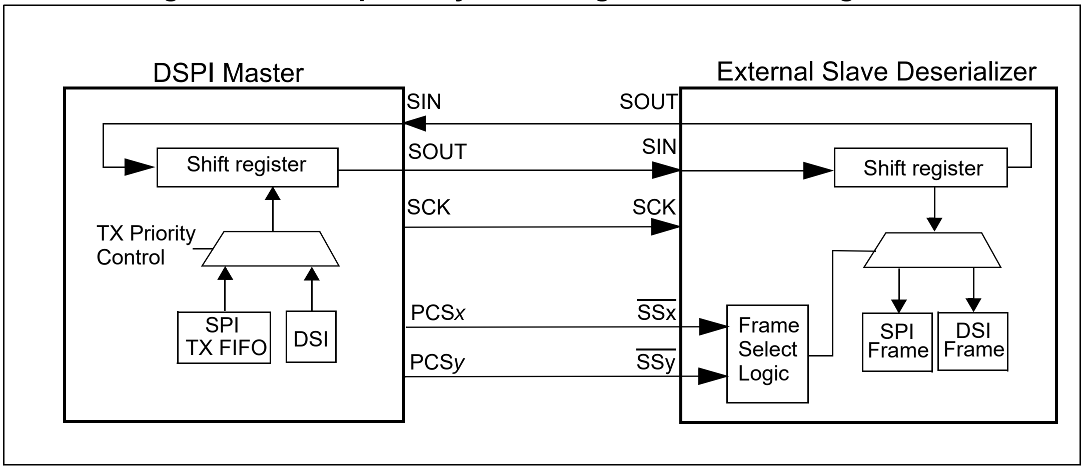
> Once slave select is asserted, there is a very small window (of up to three module clocks) after every SOF (Start of Frame) during which if a write on ASDR/SDR registers occurs, the next frame to be transmitted would be loaded into the shift register. Since it is not feasible that such a write is initiated and completed within this short span of time, the data transmitted by DSPI slave would lag by one data frame.

49.4.4 Combined Serial Interface (CSI) configuration

The CSI Configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis.

CSI Configuration allows interleaving of DSI data frames from the parallel input signals with SPI data frames from the TX FIFO. The data returned from the bus slave is either used to drive the parallel output signals or it is stored in the RX FIFO. The CSI Configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI Configuration when the DCONF field in the DSPI_MCR is 0b10. *Figure 912* shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

Figure 912. Example of System using DSPI in CSI Configuration



In CSI Configuration, the DSPI transfers DSI data based on *Section 49.4.3.5: DSI transfer initiation control*. When there are SPI commands in the TX FIFO, the SPI data has priority over the DSI frames. When the TX FIFO or CMD FIFO is empty, DSI transfer resumes.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTARs associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy.

The CSI Configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the parallel output signals. Data returned from the external slave while a SPI frame is transferred is moved to the RX FIFO. The TX FIFO, CMD FIFO and RX FIFO are fully functional in CSI mode.

49.4.4.1 CSI serialization

Serialization in the CSI configuration is similar to serialization in DSI Configuration. The transfer attributes for SPI frames are determined by the DSPI_CTAR selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI_CTAR selected by the DSICR0[DSICTAS] field.

The parallel inputs signal states are latched into the DSPI DSI Serialization Data Register (DSPI_SDR) on the rising edge of every protocol clock and serialized based on the transfer initiation control settings in the DSPI_DSICR1/0. When SPI frames are written to the TX FIFO (and when CMD FIFO is not empty), they have priority over DSI data and are transferred at the next frame boundary. A copy of the most recently transferred DSI frame is stored in the DSPI_COMPR1/0. The Transfer Priority Logic selects the source of the serialized data and asserts the appropriate PCS signal.

Separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI frames.

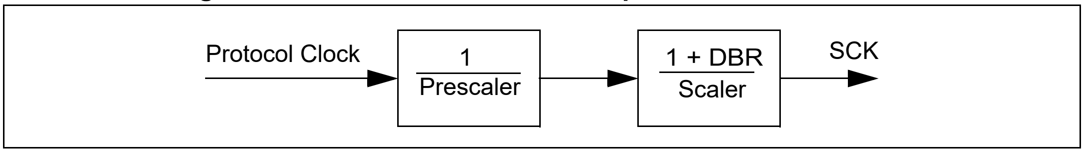
49.4.4.2 CSI deserialization

The deserialized frames in CSI Configuration goes into the DSPI_DDR1/0 or the RX FIFO based on the transfer priority logic. When DSI frames are transferred, the returned frames are deserialized and latched into the DSPI_DDR1/0. When SPI frames are transferred, the returned frames are deserialized and written to the RX FIFO.

49.4.5 DSPI baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. *Figure 913* shows conceptually how the SCK signal is generated.

Figure 913. Communications clock prescalers and scalers



49.4.5.1 Baud rate generator

The baud rate is the frequency of the Serial Communication Clock (SCK). The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. *Table 999* shows an example of how to compute the baud rate.

Table 999. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

Note: The clock frequencies mentioned in Table 999 are given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

49.4.5.2 PCS to SCK delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. The PCSSCK and CSSCK fields in the CTAR*n* registers select the PCS to SCK delay by the formula in the CSSCK field description. *Table 1000* shows an example of how to compute the PCS to SCK delay.

Table 1000. PCS to SCK delay computation example

f_P	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCSCSK and CSSCK fields have no effect in TSB configuration.

Note: The clock frequency mentioned in Table 1000 is given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

49.4.5.3 After SCK delay (t_{ASC})

The After SCK delay is the length of time between the last edge of SCK and the negation of PCS. The PASC and ASC fields in the CTAR*n* registers select the After SCK delay by the formula in the ASC field description. Table 1001 shows an example of how to compute the After SCK delay.

Table 1001. After SCK delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK delay
100 MHz	0b01	3	0b0100	32	0.96 μs

PCASC and ASC fields have no effect in TSB configuration.

Note: The clock frequency mentioned in Table 1001 is given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

49.4.5.4 Delay after transfer (t_{DT})

The delay after transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. The PDT and DT fields in the CTAR*n* registers select the Delay after Transfer by the formula in the DT field description. Table 1002 shows an example of how to compute the delay after transfer.

Table 1002. Delay after transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

Note: The clock frequency mentioned in Table 1002 is given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

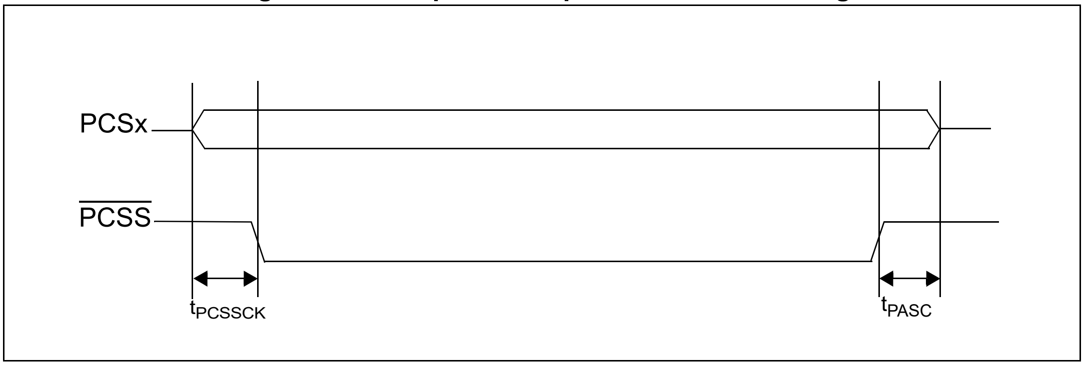
When in non-continuous clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] bitfield description. When in continuous clock mode and TSB is not enabled, the delay is fixed at 1 SCK period.

In TSB mode, the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. Refer to detailed information on Section 49.4.9: Timed Serial Bus (TSB).

49.4.5.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The PCSS signal provides a delay to allow the PCS signals to settle after a transition occurs, thereby avoiding glitches. When the DSPI is in master mode and the PCSSE bit is set in the MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode the PCS[0]–PCS[4]and PCS[6]–PCS[7] signals into as many as 128 glitch-free PCS signals. Figure 914 shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

Figure 914. Peripheral Chip Select Strobe Timing



The delay between the assertion of the PCS signals and the assertion of PCSS is selected by the PCSSCK field in the CTAR based on the following formula:

Equation 41

$$t_{PCSSCK} = \frac{1}{f_P} \times PCSSCK$$

At the end of the transfer the delay between $\overline{PCSS}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

Equation 42

$$t_{PASC} = \frac{1}{f_P} \times PASC$$

Table 1003 shows an example of how to compute the t_{PCSSCK} delay.

Table 1003. Peripheral Chip Select Strobe Assert Computation Example

f_P	PCSSCK	Prescaler	Delay before transfer
100 MHz	0b11	7	70.0 ns

Table 1004 shows an example of how to compute the t_{PASC} delay.

Table 1004. Peripheral Chip Select Strobe Negate Computation Example

f_P	PASC	Prescaler	Delay after transfer
100 MHz	0b11	7	70.0 ns

The $\overline{PCSS}$ signal is not supported when Continuous Serial Communication SCK or TSB mode is enabled.

Note: The clock frequency mentioned in the preceding Table 1003 and Table 1004 is given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

49.4.6 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTAR*n*) select the polarity and phase of the serial clock, SCK.

- CPOL: selects the idle state polarity of the SCK.
- CPHA: selects if the data on SOUT is valid before or on the first SCK edge.

Even though the bus slave does not control the SCK signal, in slave mode these values must be identical to the master device settings to ensure proper transmission. In SPI slave mode, only CTAR0 is used.

In DSI slave mode, only CTAR1 is used.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0.
- Classic SPI with CPHA = 1.
- Modified Transfer format with CPHA = 0.
- Modified Transfer format with CPHA = 1.

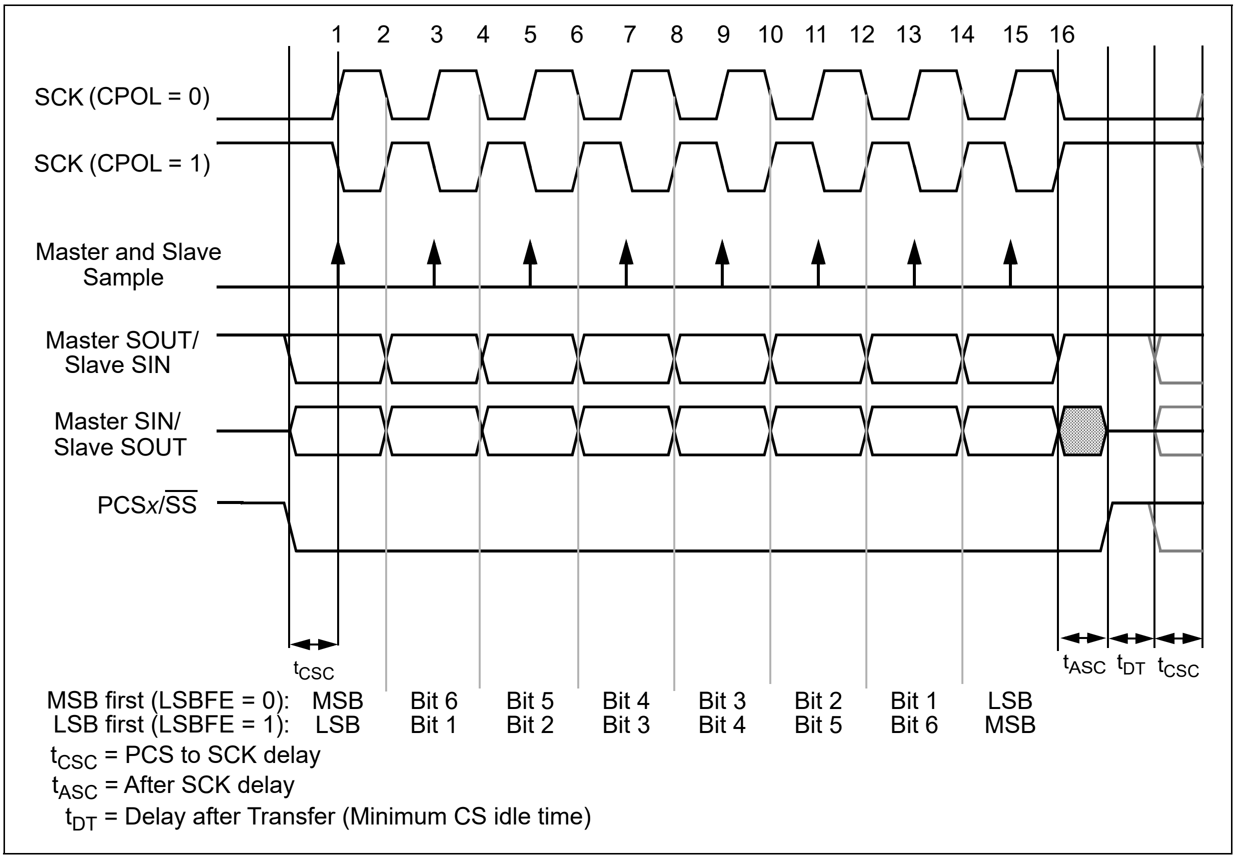
A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI and DSI configurations, the DSPI provides the option of keeping the PCS signals asserted between frames. Refer to *Section 49.4.6.5: Continuous Selection Format* for details.

49.4.6.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in *Figure 915* is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

Figure 915. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)



The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals.

At the second edge of the SCK the master and slave devices place their second data bit on their serial data output signals.

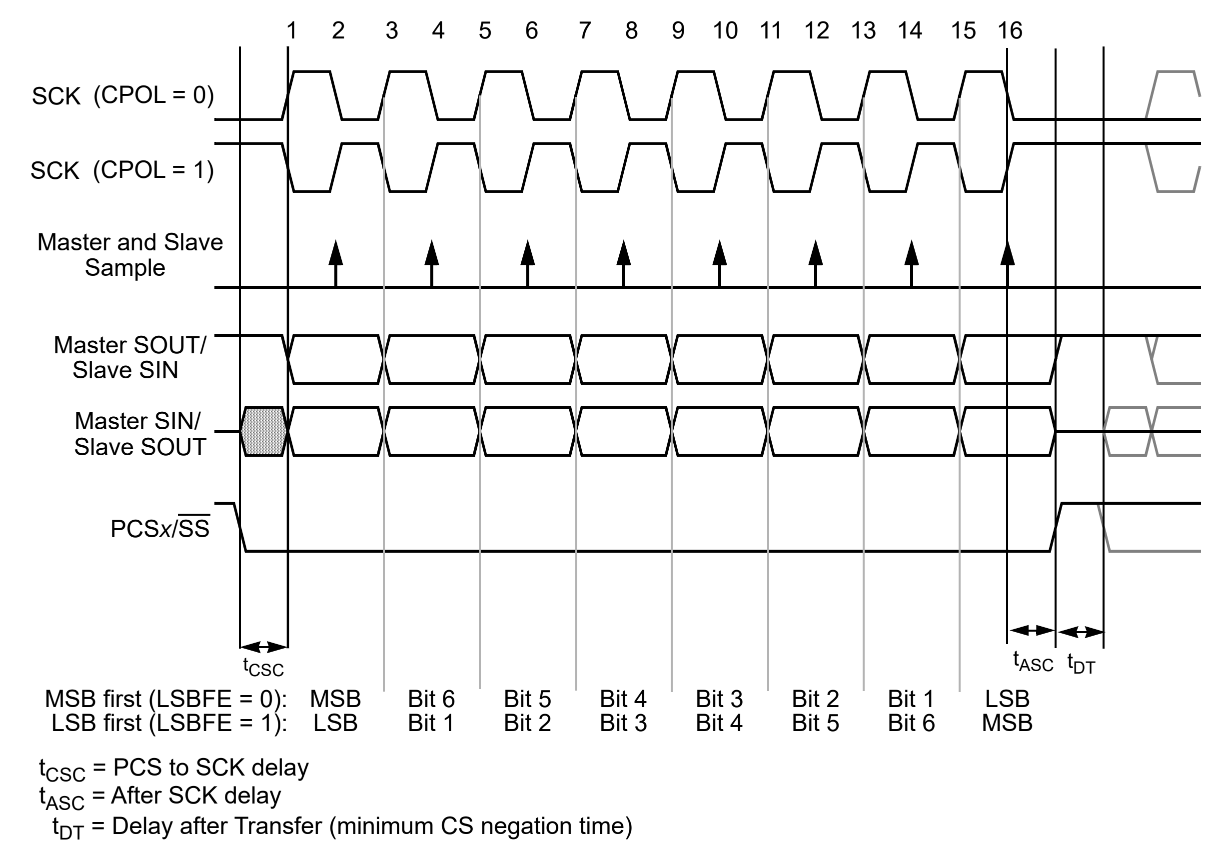
For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges.

After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

49.4.6.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in *Figure 916* is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

Figure 916. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8)



The master initiates the transfer by asserting the PCS signal to the slave.

After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins.

For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges.

After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

49.4.6.3 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actually sample point is delayed by 1 protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE = 1. Timing delays shown are:

- t_{CSC} – PCS to SCK assertion delay
- t_{ACS} – After SCK PCS negation delay
- t_{su_ms} – master SIN setup time
- t_{hd_ms} – master SIN hold time
- t_{vd_sl} – slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- t_{su_sl} – data setup time on slave data input
- t_{hd_sl} – data hold time on slave data input
- t_p – protocol clock period

Figure 917 shows the modified transfer format for CPHA = 0 and f_p/f_{sck} = 4{s}. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behaviors are shown.

- Signal, marked “SOUT of Ext Slave”, presents regular SPI slave serial output.
- Signal, marked “SOUT of DSPI Slave”, presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according to DSPI master CPHA programming.

Note: *In the following diagrams, f_p represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.*

Note: *When MTFE=1 with continuous SCK enabled (MCR [CONT_SCKE] =1) in master mode, configure CTAR[LSBFE]=0 for correct operations while receiving unequal length frames. If PUSHR[CONT] is also set for back to back frame transfer, also configure the frame size of the first frame as less than or equal to the frame size of the next frame. In this scenario, make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.*

Figure 917. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$)

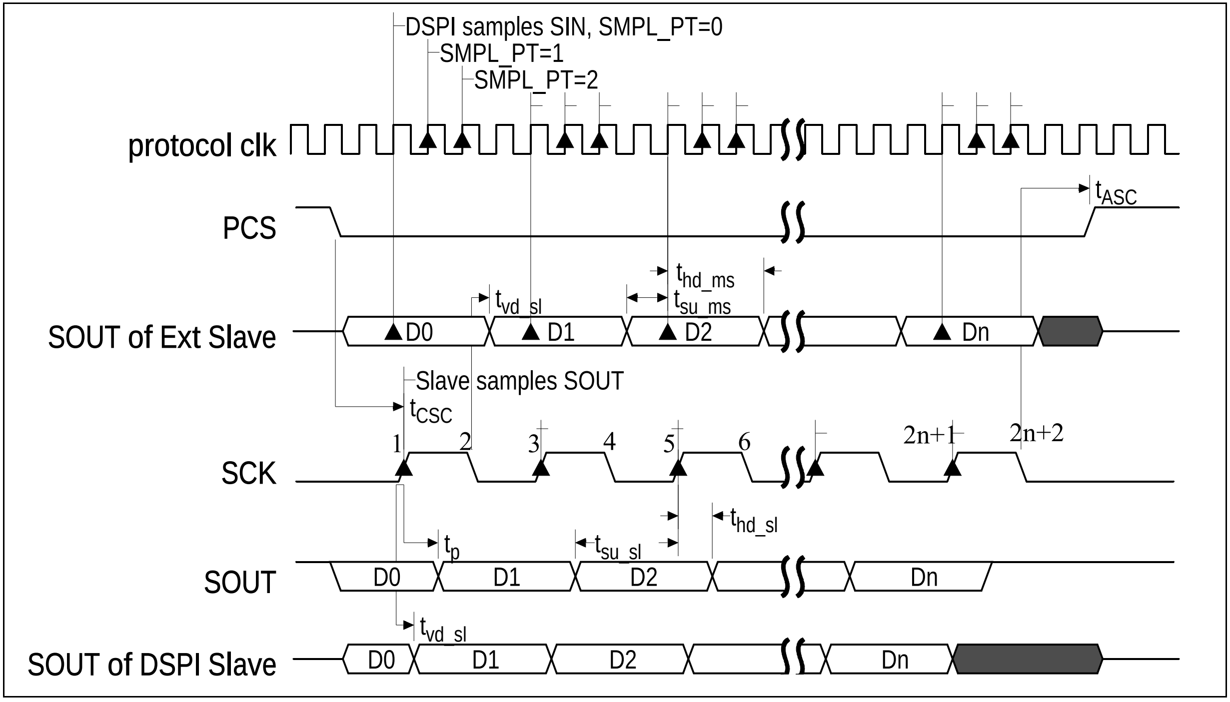


Figure 918. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$)

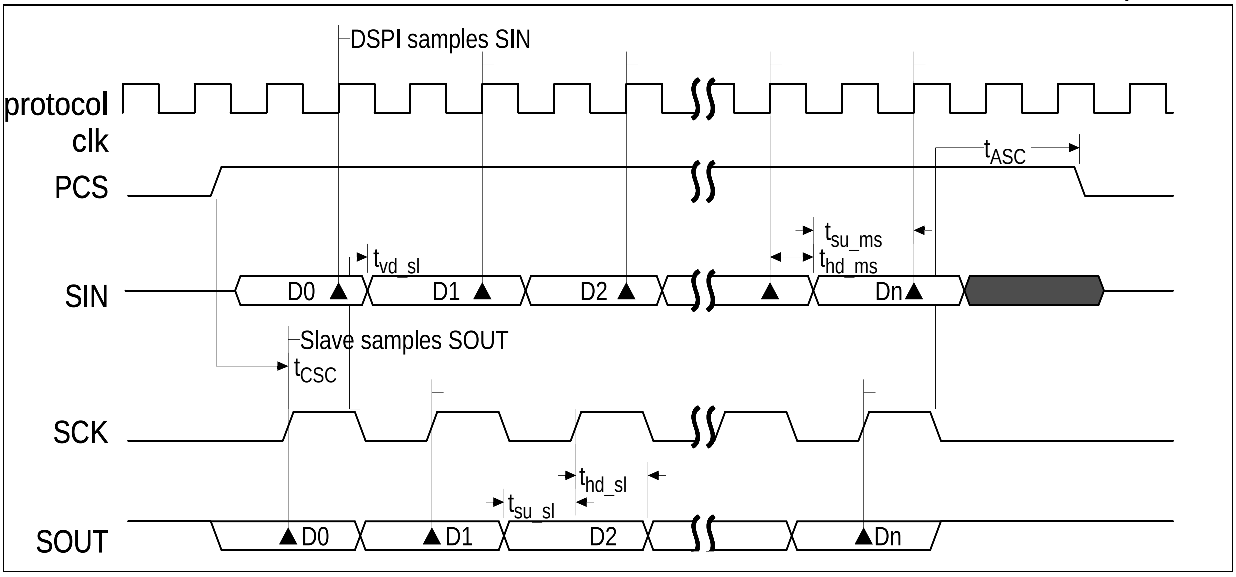
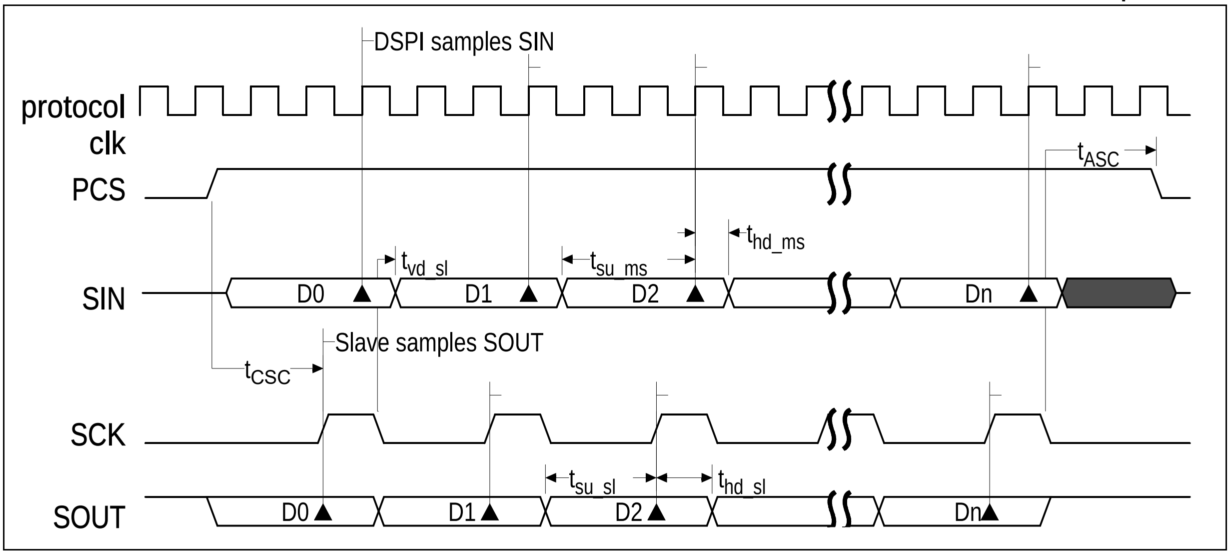


Figure 919. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$)



49.4.6.4 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. The SCK to PCS delay must be greater than or equal to half of the SCK period.

Figure 920. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$)

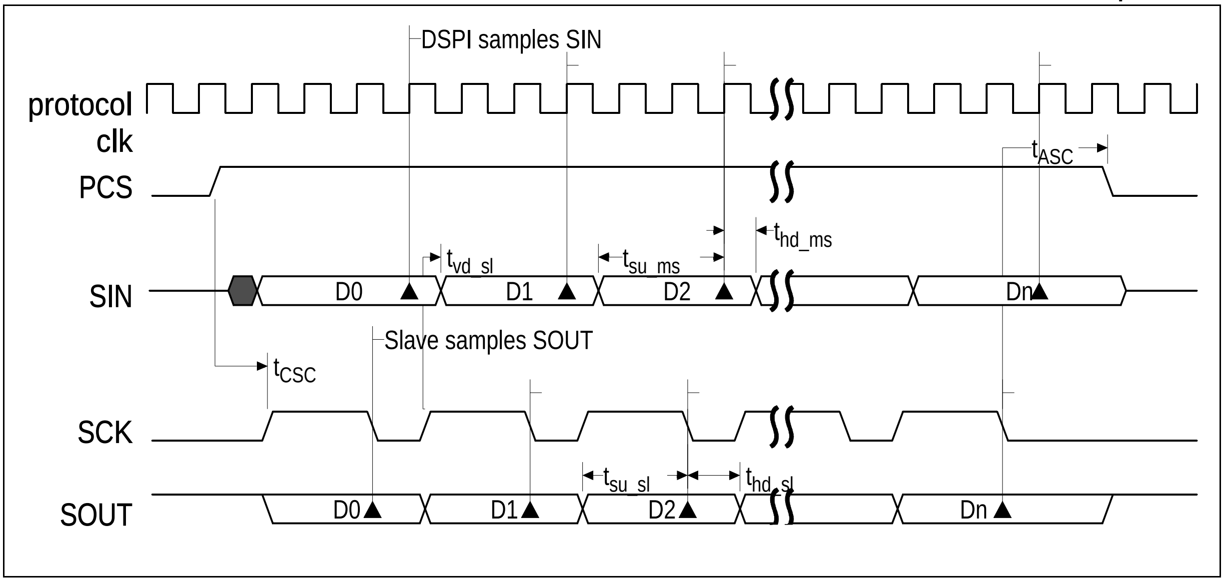


Figure 921. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$)

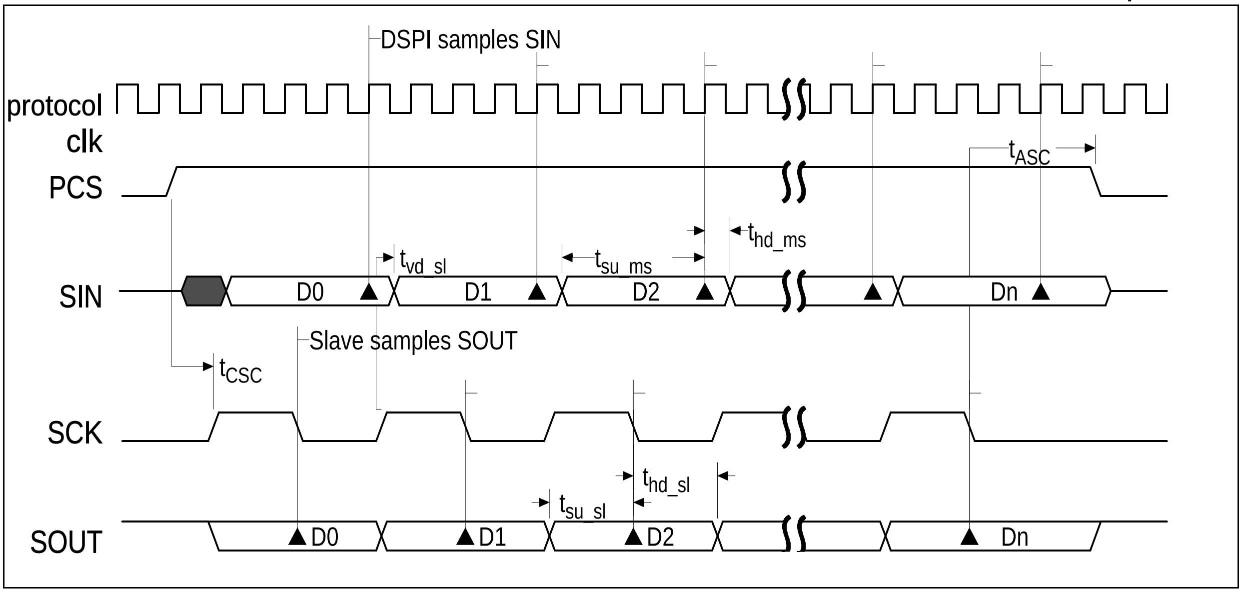
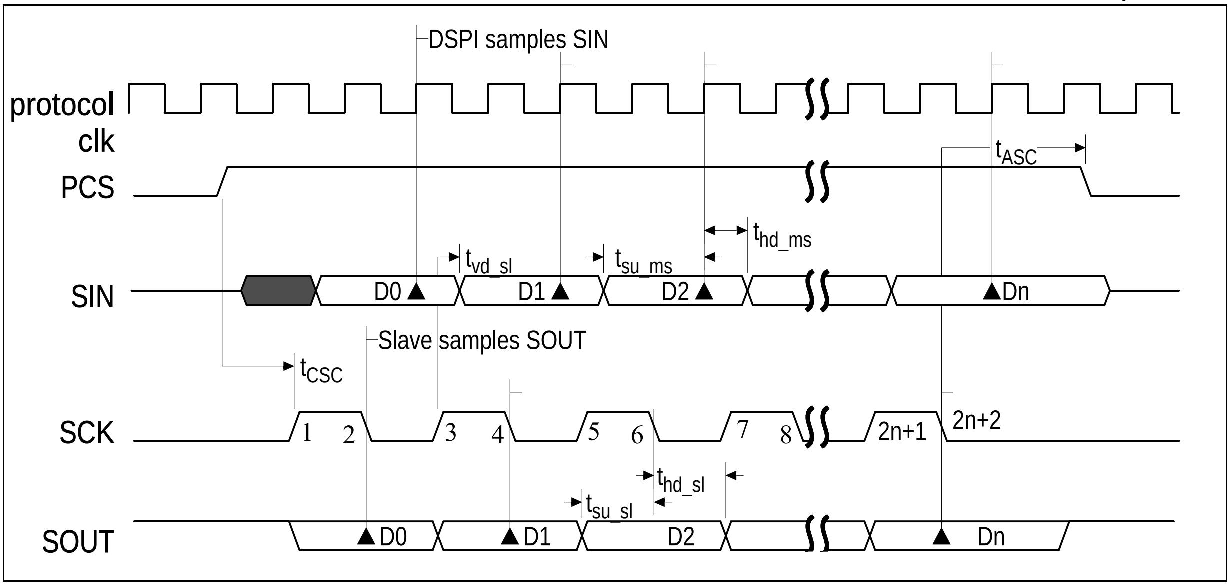


Figure 922. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$)

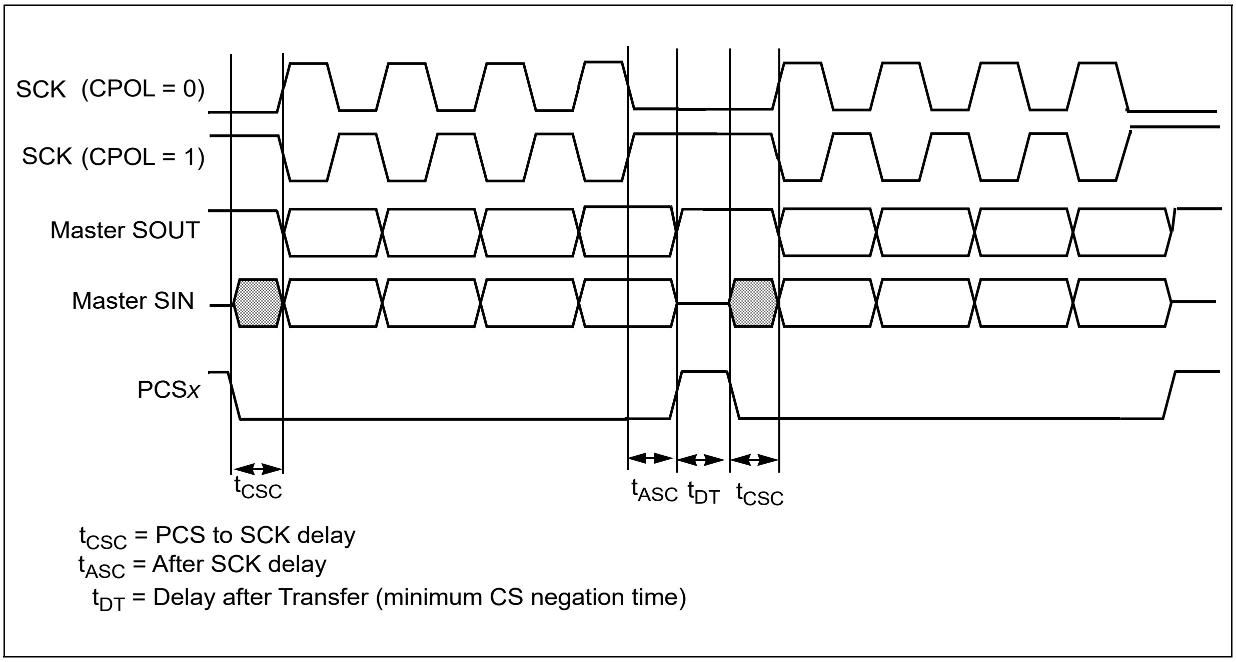


49.4.6.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both the following cases. The Continuous Selection Format is enabled for the SPI Configuration by setting the CONT bit in the SPI command. Continuous Selection is enabled for the DSI Configuration by setting the DCONT bit in the DSICR0.

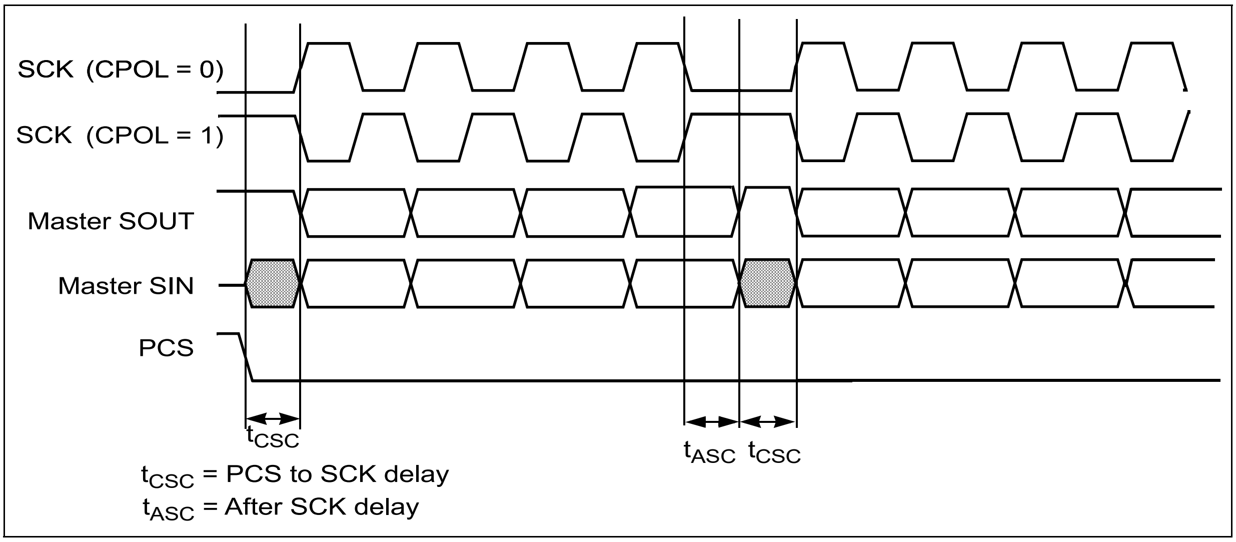
The behavior of the PCS signals in the two configurations is identical so only SPI Configuration will be described. When the CONT bit = 0, the DSPI drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSIS*n* bits in the MCR. The timing diagram in *Figure 923* is for two four-bit transfers with CPHA = 1 and CONT = 0.

Figure 923. Example of non-continuous format (CPHA = 1, CONT = 0)



When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. *Figure 924* shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

Figure 924. Example of continuous transfer (CPHA = 1, CONT = 1)



When using DSPI with continuous selection follow these rules:

- All transmit commands must have the same PCS*n* bits programming.
- The CTARs selected by transmit commands must be programed with the same transfer attributes. Only FMSZ field can be programed differently in these CTARs.
- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit de-asserted (in master mode) and the user software must provide sufficient frames in the TX_FIFO to be sent out (in slave mode) and the master de-asserts the PCS*n* at end of transmission of last frame.
- The PUSHR[CONT]/DSICR0[DCONT] bits must be de-asserted before asserting MCR[HALT] bit (in master mode). This ensures that the PCS*n* signals are deasserted. Asserting MCR[HALT] bit during continuous transfer will cause the PCS*n* signals to remain asserted and hence Slave Device cannot transition from RUNNING to STOPPED state.

Note: You must fill the TXFIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TXFIFO becomes empty.

Note: When operating in slave mode, ensure that when the last-entry in the TXFIFO is completely transmitted (that is the corresponding TCF flag is asserted and TXFIFO is empty), the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

49.4.6.6 Fast Continuous Selection Format

The Fast Continuous Selection Format functions similar to the Continuous Selection Format except that the inter command delays, t_{ASC} and t_{CSC}, can be masked out and are not inserted by the hardware.

Note: The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled. Masking of delays is not allowed in DSI and CSI configurations or if the transfer is non-continuous.

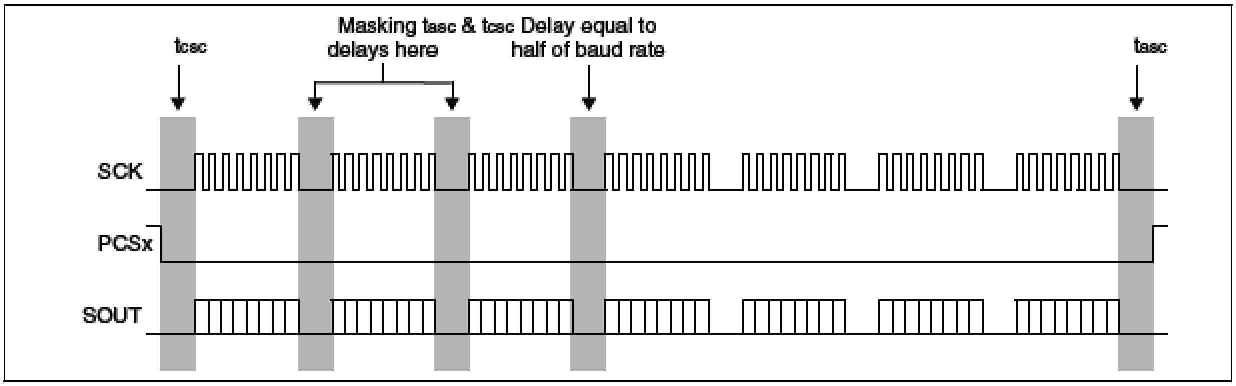
The Fast Continuous Selection Format is enabled by writing ‘1’ into FCPCS bit of the MCR. When this bit is asserted, MASC and MCSC bits of the PUSHR perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT bit asserted. In order to avoid these delays and speed up the transfer process, the software can simply mask these delays while programming the command in the PUSHR.

While masking the delays, the software must follow the following masking rules, else correct operation is not guaranteed.

- MASC bit masks the “After SCK” delay for the current frame.
- MCSC bit masks the “PCS to SCK” delay for the next frame.
- “After SCK” (t_{ASC}) delay must not be masked when the current frame is the last frame in the continuous selection format.
- The “PCS to SCK” delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here, too.

Figure 925 shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

Figure 925. Example of Fast Continuous Selection Format



In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

49.4.7 Continuous Serial Communications Clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous Serial Communications Clock regardless of the MCR[HALT] bit status. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the DSPI is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- When the DSPI is in DSI configuration, the CTAR specified by the DSICTAS field is used at all times.
- When the DSPI is in CSI configuration, the CTAR selected by the DSICTAS field is used initially.
  - At the start of a SPI frame transfer, the CTAR specified by the CTAS value for the frame is used.
  - At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

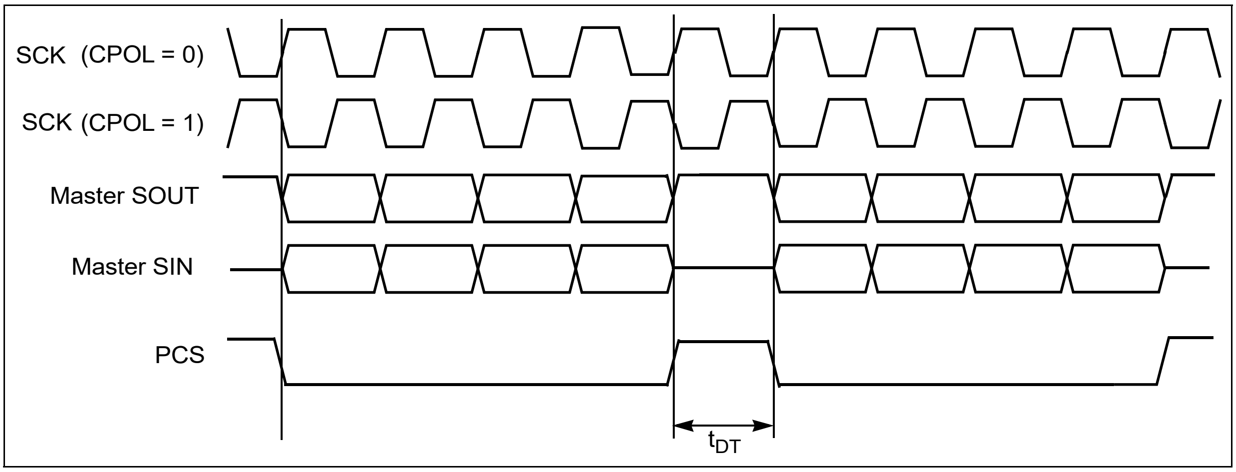
It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. When TSB configuration is enabled the t_{DT} is programmable from 1 to 64 SCK cycles.

Figure 926 is the timing diagram for Continuous SCK format with Continuous Selection disabled.

Note: When in Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TXFIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

Figure 926. Continuous SCK timing diagram (CONT = 0)



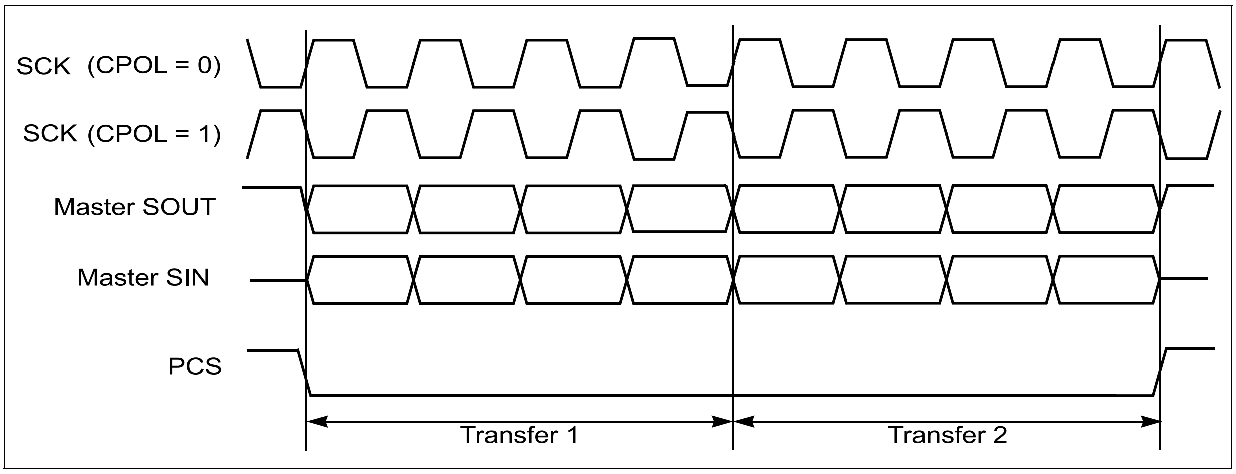
If the CONT bit in the TX FIFO entry is set or the DCONT in the DSICR0 is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with

PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO.
- Continuous SCK with CONT bit set and entering STOPPED state (refer to *Section 49.4.1: Start and Stop of DSPI transfers*).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 927 shows the timing diagram for Continuous SCK format with Continuous Selection enabled.

Figure 927. Continuous SCK Timing Diagram (CONT = 1)



49.4.8 Slave mode operation constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the DSPI is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and SS signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR1/0 register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

49.4.9 Timed Serial Bus (TSB)

The DSPI can be programed in Timed Serial Bus configuration by setting DSICR0[TSBC] bit. Refer to *Section 49.3.13: DSPI DSI Configuration Register 0 (DSPI_DSICR0)* for details.

TSB configuration provides the Micro Second Channel (MSC) downstream channel support.

The MSC upstream channel is not supported by the DSPI, but can be supported by any available Serial Communication Controller (SCI or UART) in the device.

To work in TSB mode the DSPI must be in master mode and in DSI (DCONF = 0b01) or CSI (DCONF = 0b10) configuration. Both Continuous and Non Continuous Serial Communication Clock (controlled by the DSPI_MCR[CONT_SCKE] bit) are supported in the TSB mode.

Figure 928 shows the signals used in the TSB interface.

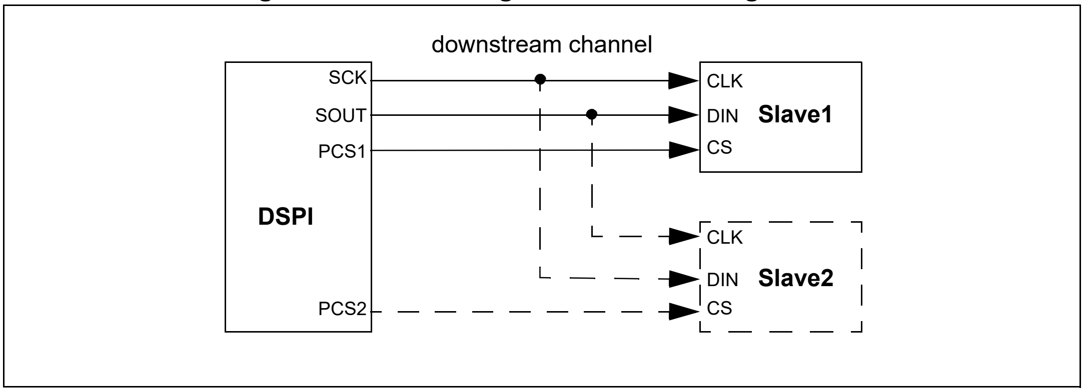
In the TSB configuration the DSPI is able to send from 4 to 66 bits MSC data frames (4 to 64 serialized data bits and up to 2 Data Selection zero bits). The serialized data bits source can be either:

- the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR1/0), written by the host software,
- parallel input pin states latched into the DSPI DSI Serialization Data Register (DSPI_SDR1/0).

DSICR0[TXSS] bit or DSPI_SSR1/0 register bits define the source of the data.

The Least Significant Bits of the DSPI_ASDR0 or DSPI_SDR0 registers are selected to be serialized if the data frame is set to less than 32 bits.

Figure 928. DSPI usage in the TSB Configuration



The PCS signals are driven together with SOUT. The t_{CSC} and t_{ASC} delays are not available. Delay after Transfer (DT) is set in SCK clock periods as a binary number formed by concatenation of the DSPI_CTAR*n* PDT and DT fields plus one, allowing to set DT from 1 to 64 serial clock periods. DT field provides least significant bits and PDT field provides most significant bits of the Delay after Transfer.

Figure 929. TSB Downstream Frames

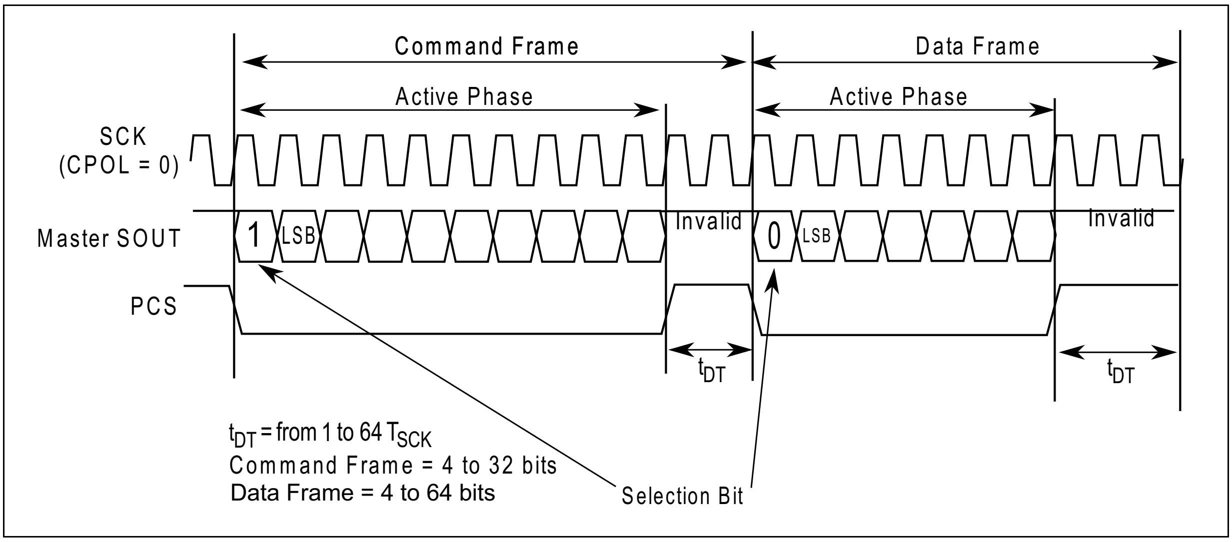


Figure 929 shows two types of MSC downstream frames: command frame, and data frame.

The first transmitted bit, called the selection bit, determines the frame type:

- The selection bit ‘0’ indicates a data frame
- The selection bit ‘1’ indicates a command frame

Data frame may contain up to 2 selection bits to support two external slave devices, (so called dual receiver configuration) or no selection bits at all.

The command frame can be written by software, through SPI TX FIFO, using one or two FIFO entries with help of the CONT bit or MCR[XSPI]. The data frame consists of up to 64 bits from the SDR1/0 or ASDR1/0 registers and up to two zero selection bits. Number of data bits in the data frame is defined by the DSCICR1[TSBCNT] field.

The selection bit of the MSC command frames (1) can be implemented by software.

The selection bits in the data frames are enabled by DSPI_DSICR1 DSE0 and DSE1 bits. Each DSE*n* bit set increases the data frame size by one bit.

To comply with MSC specification, set DSPI_CTAR*n*[LSBFE] to transmit least significant bit first.

Regardless of the LSBFE bit setting, the data frame selection bits, if enabled, are always transmitted first, before the corresponding data subframes.

49.4.9.1 MSC Dual Receiver Support with PCS Switchover

When in TSB mode it is possible to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame. The bit, at which this switchover occurs, is defined by FMSZ field of the DSPI_CTAR*n* register, selected by DSICTAS field of the DSICR0 register. If DSICR1[DSI64E] is enabled, then the bit at which the PCS switch occurs, is defined by the concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields.

Note: When using Dual receiver support, it must be ensured that the number of bits to be transmitted to each slave must be at least 4.

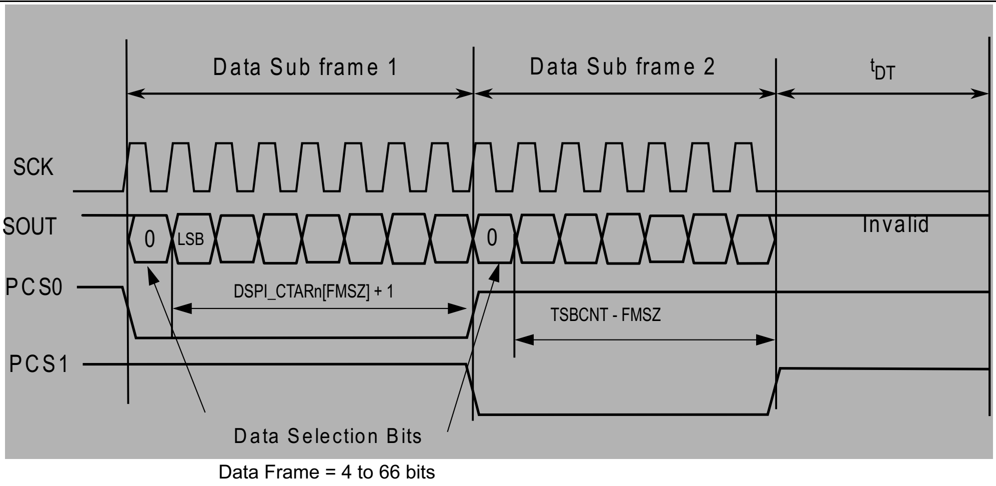
Number of the bits, not including the Data Selection Bit, in the first part of the frame is equal to value of the FMSZ field plus one. (This value changes to {FMSZ4,FMSZ} plus one, when DSICR1[DSI64E] is enabled). During this part of the frame the PCS signal levels are controlled by DSPI_DSICR0 DPCSn bits, after that by DSPI_DSICR1 DPCS1_n bits.

The PCS switchover occurs at the driving edge of the SCK clock output.

The second Data Selection Bit is inserted after the PCS switchover if enabled.

The Data Frame with PCS switchover is shown in *Figure 930*.

Figure 930. TSB Data Frame Format for MSC Dual Receiver Operation



49.4.10 Interleaved TSB (ITSB) Mode

This mode is similar to the TSB in all aspects except for the differences mentioned in *Table 1005*. In the Interleaved TSB or ITSB mode, frame transmission is governed by the following rules:

- Transmission of frames is executed on a periodic trigger that is generated internally to the DSPI. The trigger period is calculated as follows:
  - Trigger Period = 64 bits (maximum data frame size) + 2 selection bits (maximum bits in data frame) + t_{PT}

    t_{PT} is the passive frame time or dead time inserted to have constant trigger period.
  - The user software shall configure the trigger period and trigger source.
- If the previously transmitted frame was a SPI frame or the TX FIFO (or CMD FIFO) is empty, the frame transmitted on subsequent trigger will always be a DSI frame.
- If the previously transmitted frame was not a SPI frame and the TX_FIFO (and CMD FIFO) is not empty, the frame transmitted on subsequent trigger will always be a SPI frame.

This mode does not change the way DSI and SPI frame transmissions are done except that ITSB mode interleaves the source (SPI or DSI) in such a way to prevent back to back SPI frames. All features of SPI and DSI mode can be used except for the Continuous Frame Selection format (PUSHR[CONT] and DSICR0[DCONT] bits cannot be asserted).

MCR[MTFE] should be set to 0 during ITSB Mode of operation, since MicroSecond Channel upstream operation is not supported.

Table 1005 lists the differences between the TSB and ITSB modes.

Table 1005. Differences between TSB and ITSB modes

Mode attribute	TSB mode	ITSB mode
Frame priority	Priority available to frames from SPI. Back to back SPI frames are sent out.	No priority. Messages are interleaved such that there are no back-to-back SPI frames
Frame selection bit	Selection bit inserted for DSI frames only. Selection bit for SPI frames is inserted by software.	Selection bit for both DSI and SPI frames inserted automatically. Encoding of selection bit same as TSB mode (0 = DSI; 1 = SPI).
Frame transmission	Occasionally triggered transmission (that is trigger is received via hardware trigger input).	Periodically triggered transmission (that is frames are transmitted continuously on every trigger).
Frame gap	Gap between frames is configurable. Time between frames is t_{DT} which is configurable from 1 to 64 baud rate clock cycles.	Gap between frames is inserted automatically as the whole operation is based on a periodic trigger. If a frame has less than 64-bits to be transmitted, dead or passive time is automatically inserted by waiting for the next trigger pulse. Time t_{DT} is not used in this mode as it can be made part of the trigger period (as passive time).

In TSB and ITSB mode, separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI. MSC dual receiver support with PCS switchover is also supported in ITSB mode. Refer to *Section 49.4.9.1: MSC Dual Receiver Support with PCS Switchover* for more details.

The ITSB mode is suitable to implement the Downstream Channel for the MSC Bus because of the ITSB mode features listed in the table above.

49.4.10.1 Configuring DSPI for ITSB mode

The following steps give the recommended way to initialize the DSPI for the ITSB mode of operation:

- The DSPI should be put in HALT mode (by asserting the MCR[HALT] bit) before programming the registers.
- Set the DSICR0[TSBC] and DSICR0[ITSB] bits. The DSICR0[TRRE] and DSICR0[CID] bits should be cleared. DSICR0[DCONT] should not be used in ITSB mode. If DSICR0[ITSB] bit is set without setting DSICR0[TSBC], then the DSPI operates in the Normal Mode depending on MCR[DCONF] bit.
- Enable frame completion interrupts for either any frame completion (TCF) or separate frame completion (DSITCF and SPITCF) as required by application.
- When remainder configuration of DSPI is complete, clear the MCR[HALT] bit to allow DSPI to start operations.

Once configured and DSPI is brought out of HALT, the ITSB mode shall start sequencing the frames from either DSI or SPI as per the rules mentioned in *Section 49.4.10: Interleaved TSB (ITSB) Mode*.

49.4.10.2 Using ITSB mode for MSC downstream channel

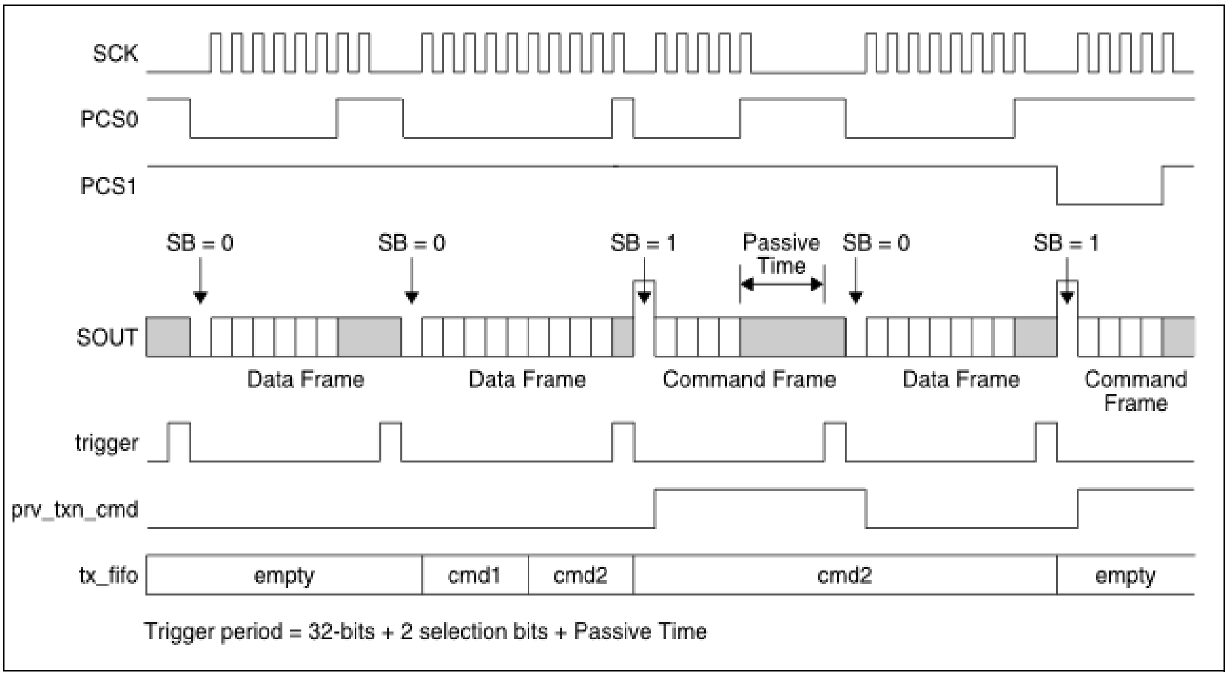
MSC Downstream Channel uses two types of frames: command frames and data frames. The Downstream operation of these frames require:

- Data frames can be sent back to back.
- Back to back command frames are not a legal scenario.
- If command frame has less bits than data frame, a gap is inserted to ensure next data frame starts at known reference time.
- In dual MSC ASIC, some bits of the data frame (which can be up to 64 bits) goes to first ASIC and remainder bits to second ASIC.
- In dual MSC ASIC, if a command frame is sent to one ASIC, it must not change the timing of the next data frame.

The ITSB mode can help implement the MSC Downstream Channel by allowing the command frames to be sent via the SPI interface and data frames to be sent via the DSI interface. Once the DSPI has been configured to work in ITSB mode (Refer to *Section 49.4.10.1: Configuring DSPI for ITSB mode*), the user software can program the DSI and SPI sides to send data and command frames respectively. The programming of frames for transmission via DSI or SPI is same as explained in *Section 49.4.4: Combined Serial Interface (CSI) configuration*.

Figure 931 shows a sample MSC Downstream transmission using the ITSB mode.

Figure 931. Sample MSC downstream transmission using ITSB mode



Brief description of operation includes:

- At the first trigger, since there are no command frames, a data frame is transmitted: to send a command frame at the start of operation, the user software must program a command frame in the TX_FIFO before enabling DSPI (MCR[HALT] = 0).
- The second trigger sets off the transmission of another data frame.
- Between the second and third trigger, two command frames are pushed into the TX_FIFO.
- At third trigger, a command frame is transmitted as the TX_FIFO is not empty: since only 16 bits or less are transmitted, no transmission happens till next trigger. This is marked as the “Passive Time”. The passive time is added in all frames after frame transmission completes and until next trigger pulse is detected.
- At fourth trigger, since the previous transmission was a command frame, a data frame is transmitted.
- At fifth trigger, since one command frame is pending in TX_FIFO and last transmitted frame was not a command frame, a command frame is transmitted.
- At sixth trigger (not shown), the process continues.

49.4.11 Parity generation and check

The DSPI module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/control is done on frame basis. The register fields setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The DSPI module can be programed to stop SPI or/and DSI frame transmission in case of a frame reception with parity error.

49.4.11.1 Parity for SPI frames

When the DSPI is in the master mode the parity generation is controlled by PE and PP bits of the TX FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the DSPI stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode, the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

49.4.11.2 Parity for DSI frames

When DSPI is in Master Mode, parity generation is controlled by PE and PP bits of the DSPI_DSICR0 register similar to the SPI frames. The parity is calculated and checked for each DSI frame. DSPI_DSICR0[DCONT] bit has no effect on parity generation. In slave mode, the parity is controlled by the PE and PP bits of the CTAR1 register.

If the parity error occurs for received DSI frame, the DSPI_SR[DPEF] bit is set. If DSPI_DSICR0[PES] bit is set, the DSPI stops DSI frames transmission. To resume DSI operation, clear the DSPI_SR[DPEF] or the DSPI_DSICR0[PES] bits.

49.4.12 Interrupts/DMA requests

Certain DSPI conditions can generate only interrupt requests and other DSPI conditions can generate either interrupt requests or DMA requests. *Table 1006* lists these conditions.

Table 1006. Interrupt and DMA request conditions

DSPI condition	Flag	Request type	
		Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	—
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	—
Transfer Complete	TCF	Yes	—
CMD Transfer Complete	CMDTCF	Yes	—
SPI Transfer Complete	SPITCF	Yes	—
DSI Transfer Complete	DSITCF	Yes	—
TX FIFO Underflow	TFUF	Yes	—
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	—
SPI Parity Error	SPEF	Yes	—
DSI Parity Error	DPEF	Yes	—
DSI Deserialized Data Match	DDIF	Yes	Yes

Each condition has a flag bit in the DSPI Status Register (SR) and a Request Enable bit in the DSPI DMA/Interrupt Request Select and Enable Register (RSER). The TX FIFO Fill Flag (TFFF) and RX FIFO Drain Flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the RSER.

The DSPI module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

49.4.12.1 End of Queue interrupt request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI command is set and the EOQF_RE bit in the RSER is set.

When MCR[XSPI] is enabled and the EOQ bit in the executing SPI command is set, the End of Queue Request will only be generated once the last Data frame in the Command Cycle has been transmitted.

Note: This interrupt request is generated when the last bit of the SPI frame with EOQ bit set is transmitted.

49.4.12.2 Transmit FIFO Fill interrupt or DMA request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

Note: SR[TFFF] flag clears automatically when DMA is used to fill TXFIFO.

Note: To clear SR[TFFF] when not using DMA, follow these steps for every PUSH performed using CPU to fill TXFIFO:

Wait until SR[TFFF] = 1

Write data to PUSHR using CPU.

Clear SR[TFFF] by writing a '1' to its location. If TXFIFO is not full, this flag will not clear.

49.4.12.3 Command FIFO Fill interrupt or DMA request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled and hence TX FIFO and CMD FIFO can be filled independently. If MCR[XSPI] is disabled, then 'TX FIFO Fill Interrupt or DMA Request' will suffice to fill both FIFO's since both FIFO's must be filled simultaneously.

Note: SR[CMDFFF] flag clears automatically when DMA is used to fill CMD FIFO.

Note: To clear SR[CMDFFF] when not using DMA, follow these steps for every PUSH performed using CPU to fill CMD FIFO:

Wait until SR[CMDFFF] = 1

Write data to Command field of PUSHR using CPU.

Clear SR[CMDFFF] by writing a '1' to its location. If CMD FIFO is not full, this flag will not clear.

49.4.12.4 Transmit FIFO Invalid Write interrupt request

The Transfer Fifo Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transfer Fifo Invalid Write Request is generated for the above condition when TFIWF_RE bit is set in the RSER.

49.4.12.5 Transfer Complete interrupt request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

49.4.12.6 Command Transfer Complete interrupt request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when the CMDTCF_RE bit is set in the RSER.

49.4.12.7 SPI Transfer Complete interrupt request

The SPI Transfer Complete Request indicates the end of transfer of an SPI frame in any Transmission Mode which uses DSPI in CSI Mode. The SPI Transfer Complete Request is generated when SR[SPITCF] flag asserts and SPITCF_RE bit is set in the RSER.

49.4.12.8 DSI Transfer Complete interrupt request

The DSI Transfer Complete Request indicates the end of transfer of a DSI frame in any Transmission Mode which uses DSPI in CSI Mode. The DSI Transfer Complete Request is generated when SR[DSITCF] flag asserts and DSITCF_RE bit is set in the RSER.

49.4.12.9 Transmit FIFO Underflow interrupt request

The Transmit FIFO Underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the DSPI, operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

49.4.12.10 Receive FIFO Drain interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

Note: SR[RFDF] flag clears automatically when DMA is used to drain RXFIFO.

Note: To clear SR[RFDF] when not using DMA, follow these steps for every POP performed using CPU to drain RXFIFO:

Wait until SR[RFDF] = 1

Read data from POPR using CPU.

Clear SR[RFDF] by writing a ‘1’ to its location. If RXFIFO is not empty, this flag will not clear.

Note: Clearing SR[RFDF] by writing a ‘1’ to its location after performing MCR[CLR_RXF], will not happen if RXFIFO is not empty because the data from shift register can be transfered just after clear operation is performed.

49.4.12.11 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

49.4.12.12 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

49.4.12.13 DSI Frame Parity Error Interrupt Request

The DSI Frame Parity Error Flag indicates that a DSI frame with parity error has been received. The DPEF_RE bit in the DSPI_RSER must be set for the interrupt request to be generated.

49.4.12.14 Deserialized Data Match Interrupt or DMA Request

The Deserialized Data Match Flag (DDIF) indicates that a DSI frame matches DSPI_DPIR1/0 data, masked with DSPI_DIMR1/0, has been received. This Request is generated if DDIF_RE bit in the DSPI_RSER is set. The DDIF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

49.4.13 Power saving features

The DSPI supports the following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory-mapped logic

Note: When in Slave Mode, the module can only be put in Low Power Modes (such as STOP and HALT) once the Master module has transitioned to the Low Power Mode as well.

49.4.13.1 Stop mode (External Stop Mode)

The DSPI supports the stop mode protocol. When a request is made to enter External Stop mode, the DSPI block acknowledges the request. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it is ready to have its clocks shut off. While the clocks are shut off, the DSPI memory-mapped logic is not accessible. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

If a DSPI block is configured in Slave mode, External Stop mode requests must be run after the last clock cycle or Low-power mode is not entered.

49.4.13.2 Module disable mode

Module disable mode is a block-specific mode that the DSPI can enter to save power. The host CPU can initiate the module disable mode by setting the MDIS bit in the MCR. The module disable mode can also be initiated by hardware.

When the MDIS bit is set, the DSPI negates Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, DSPI is said to have entered Module Disable Mode. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state.

If implemented, the Clock Enable signal can stop the clock to the non-memory-mapped logic. When Clock Enable is negated, the DSPI is in a dormant state, but the memory-mapped registers are still accessible.

Certain read or write operations have a different effect when the DSPI is in the module disable mode:

- Reading the RX FIFO Pop register does not change the state of the RX FIFO.
- Writing to the FIFO Push register does not change the state of the TX FIFO or CMD FIFO.
- Clearing of the FIFOs has no effect.
- Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect.
- All status bits and register flags in the DSPI return the correct values when read, but writing to them has no effect.
- Writing to the TCR during module disable mode has no effect.
- Interrupt and DMA request signals cannot be cleared.

49.5 Initialization/application information

This section describes how to initialize the DSPI module.

49.5.1 Managing DSPI queues

The queues are not part of the DSPI, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI Configuration.

1. When DSPI executes last command word from a queue, the EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is cleared to indicate the STOPPED state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a ‘1’ to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a ‘1’ to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

49.5.2 Switching master and slave mode

When changing modes in the DSPI, follow the steps below to guarantee proper operation.

1. Halt the DSPI by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a ‘1’ to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable the DSPI by clearing MCR[HALT].

49.5.3 Initializing DSPI in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the DSPI is enabled by clearing MCR[HALT]. It should be ensured that DSPI Slave is enabled before enabling DSPI Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

49.5.4 Baud rate settings

Table 1007 shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is clear.

Note: The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the Clocking chapter for the frequency used to drive this module in the device.

Table 1007. Baud rate values (bps)

Baud Rate scaler values (DSPIn_CTAR[BR])	Baud rate divider prescaler values (DSPI*n*_CTAR[PBR]) 2	3	5	7
2	25.0 MHz	16.7 MHz	10.0 MHz	7.14 MHz
4	12.5 MHz	8.33 MHz	5.00 MHz	3.57 MHz
6	8.33 MHz	5.56 MHz	3.33 MHz	2.38 MHz
8	6.25 MHz	4.17 MHz	2.50 MHz	1.79 MHz
16	3.12 MHz	2.08 MHz	1.25 MHz	893 kHz
32	1.56 MHz	1.04 MHz	625 kHz	446 kHz
64	781 kHz	521 kHz	312 kHz	223 kHz
128	391 kHz	260 kHz	156 kHz	112 kHz
256	195 kHz	130 kHz	78.1 kHz	55.8 kHz
512	97.7 kHz	65.1 kHz	39.1 kHz	27.9 kHz
1024	48.8 kHz	32.6 kHz	19.5 kHz	14.0 kHz
2048	24.4 kHz	16.3 kHz	9.77 kHz	6.98 kHz
4096	12.2 kHz	8.14 kHz	4.88 kHz	3.49 kHz
8192	6.10 kHz	4.07 kHz	2.44 kHz	1.74 kHz
16384	3.05 kHz	2.04 kHz	1.22 kHz	872 Hz
32768	1.53 kHz	1.02 kHz	610 Hz	436 Hz

49.5.5 Delay settings

Table 1008 shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (t_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

Note: The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the Clocking chapter for the frequency used to drive this module in the device.

This table does not apply for TSB Continuous mode.

Table 1008. Delay values

		Delay prescaler values (DSPIn_CTAR[PBR])			
		1	**3**	**5**	**7**
Delay scaler values	**2**	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

49.5.6 Calculation of FIFO pointer addresses

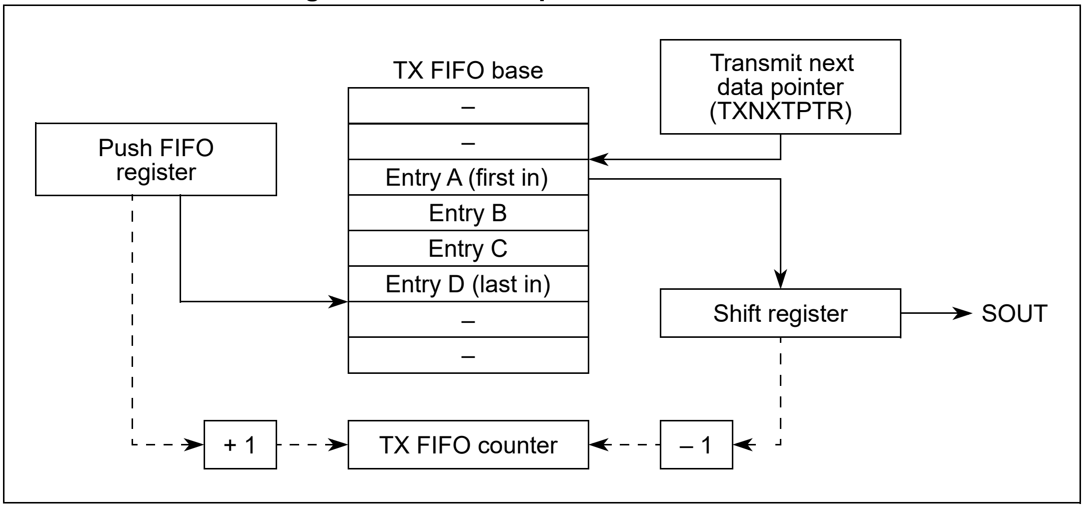
Complete visibility of the TX and CMD and RX FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and a memory-mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory-mapped.

- For TX FIFO, the first-in pointer is the Transmit Next Pointer (TXNXTPTR).
- For CMD FIFO, the first-in pointer is the Command Next Pointer (CMDNXTPTR).
- For RX FIFO, the first-in pointer is the Pop Next Pointer (POPNXTPTR).

Figure 932 illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts are applicable to the CMD FIFO and RX FIFO.

Refer to *Section 49.4.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*, *Section 49.4.2.5: Command First In First Out (CMD FIFO) buffering mechanism* and *Section 49.4.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism* for details on the FIFO operation.

Figure 932. TX FIFO pointers and counter



49.5.6.1 Address calculation for the first-in entry and last-in entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

Equation 43

$$\mathrm{First-inEntryAddress} = \mathrm{TXFIFOBase} + (4 \times \mathrm{TXNXTPTR})$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

Equation 44

$$\mathrm{Last-inEntryAddress} = \mathrm{TXFIFOBase} + 4 \times (\mathrm{TXCTR} + \mathrm{TXNXTPTR} - 1)\mathrm{mod}(\mathrm{TXFIFOdepth})$$

TX FIFO Base – Base address of TX FIFO

TXCTR – TX FIFO Counter

TXNXTPTR – Transmit Next Pointer

TX FIFO Depth – Transmit FIFO depth, implementation specific

49.5.6.2 Address calculation for the first-in entry and last-in entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

Equation 45

$$\mathrm{First-inEntryAddress} = \mathrm{CMDFIFOBase} + (4 \times \mathrm{CMDNXTPTR})$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

Equation 46

$$\mathrm{Last-inEntryAddress} = \mathrm{CMDFIFOBase} + 4 \times (\mathrm{CMDCTR} + \mathrm{CMDNXTPTR} - 1) \mathrm{mod}(\mathrm{CMDFIFOdepth})$$

CMD FIFO Base – Base address of CMD FIFO
CMDCTR – CMD FIFO Counter
CMDNXTPTR – Command Next Pointer
CMD FIFO Depth – Command FIFO depth, implementation specific

49.5.6.3 Address calculation for the first-in entry and last-in entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

Equation 47

$$\mathrm{First-inEntryAddress} = \mathrm{RXFIFOBase} + (4 \times \mathrm{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

Equation 48

$$\mathrm{Last-inEntryAddress} = \mathrm{RXFIFOBase} + 4 \times (\mathrm{RXCTR} + \mathrm{POPNXTPTR} - 1) \mathrm{mod}(\mathrm{RXFIFOdepth})$$

RX FIFO Base – Base address of RX FIFO
RXCTR – RX FIFO counter
POPNXTPTR – Pop Next Pointer
RX FIFO Depth – Receive FIFO depth, implementation specific

50 LINFlexD

50.1 Introduction

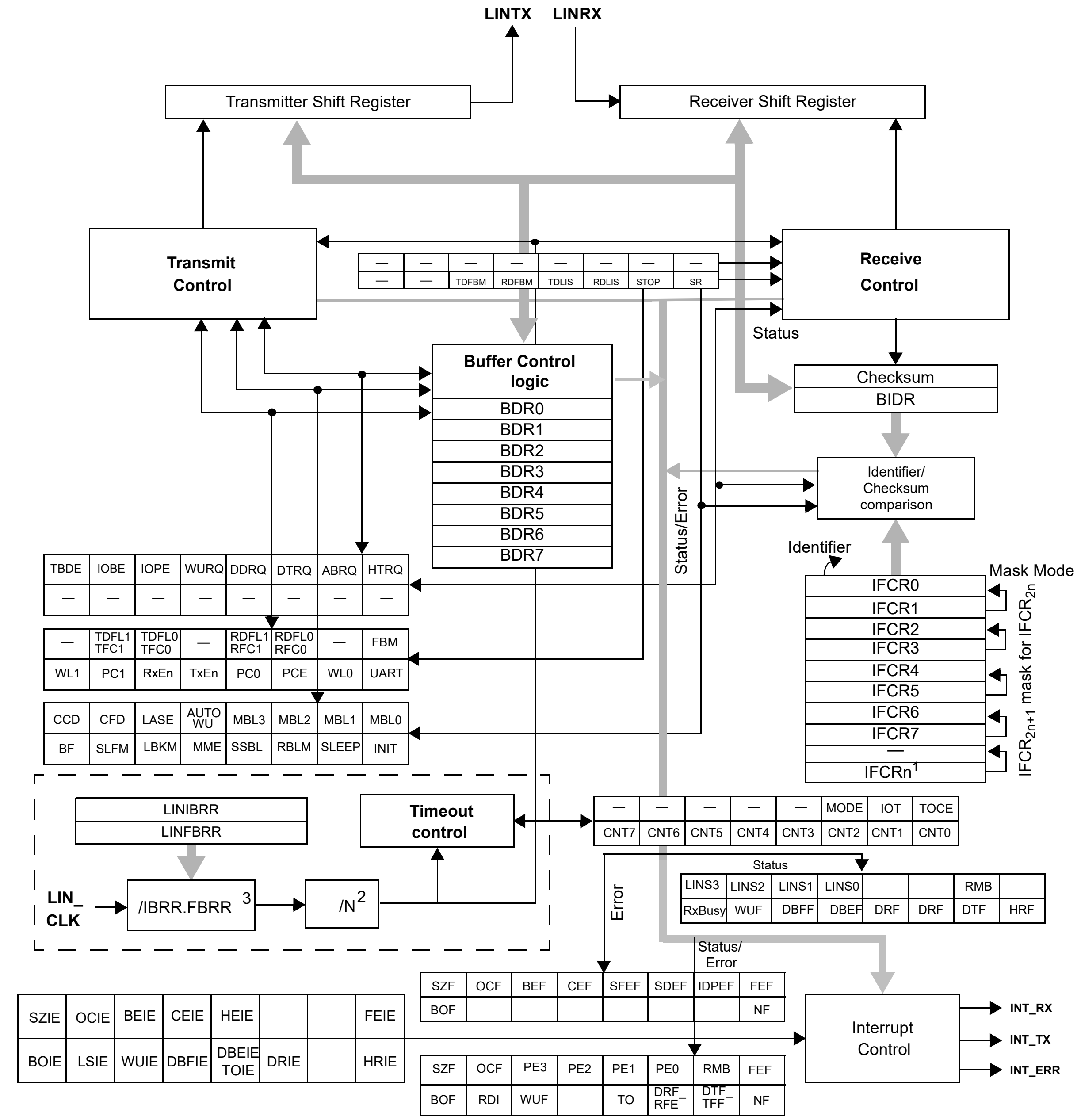
The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, 2.1 and 2.2. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/UART operating mode.

Figure 933. Block diagram



Notes:

1. The value of n depends on the no_of_filters. Refer to the device configuration chapter to see the number of filters used in this device.
2. For LIN mode N = 16 and N is programmable by user in UART mode.
3. FBRR is not usable when reduced over-sampling is enabled in UART mode.

50.2 Main features

LINFlexD controller can operate in several modes, each of which has a distinct set of features which are described in the following sections. In addition, LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
  - Initialization
  - Normal
  - Sleep
- Test mode: Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

50.2.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, and 2.1
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection
- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation (The relationship “(2/3)* LIN_CLK ≥ PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained)

50.2.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers (refer to *Section 50.3.2.8: Baud rate generation*)
- Separate clock for baud rate calculation (The relationship “(2/3)* LIN_CLK ≥ PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained)
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management

- The maximum baud rate achievable is 25 Mbit/s.

- For bit rate (BR) <= LIN_CLK/16
  - 16 times oversampling
  - 3:1 majority voting
- For LIN_CLK/16 < BR <= LIN_CLK/8
  - OSR = 8
  - 3:1 majority voting
- For LIN_CLK/8 < BR <= LIN_CLK/4
  - OSR = 4,5,6
  - 1:1 voting

50.2.3 Standard features

DMA2x bus interface.

50.3 Functional description

50.3.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task - all other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

50.3.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

Figure 934. Frames

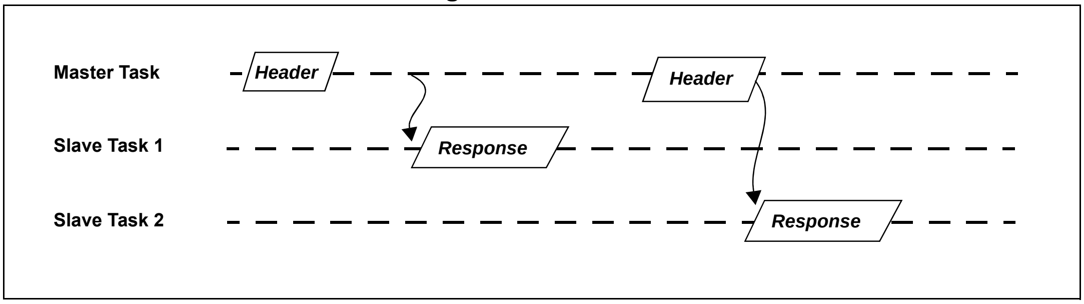
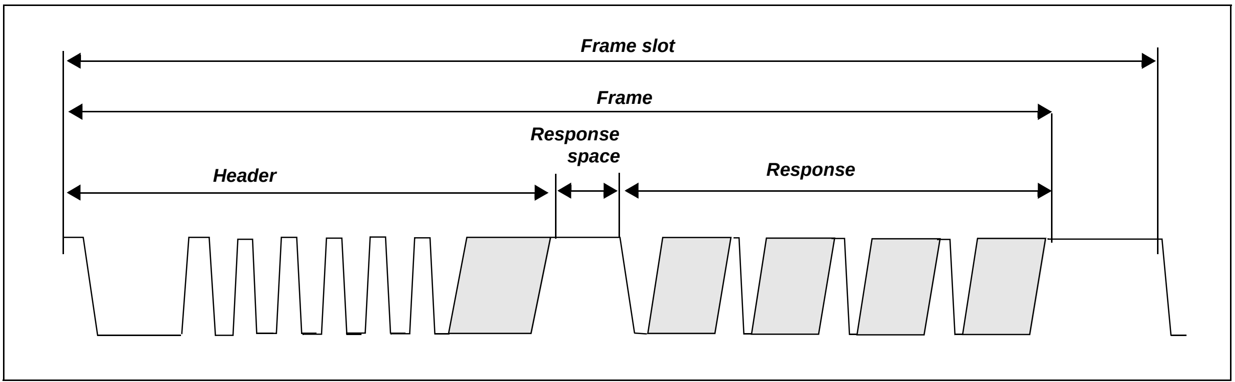


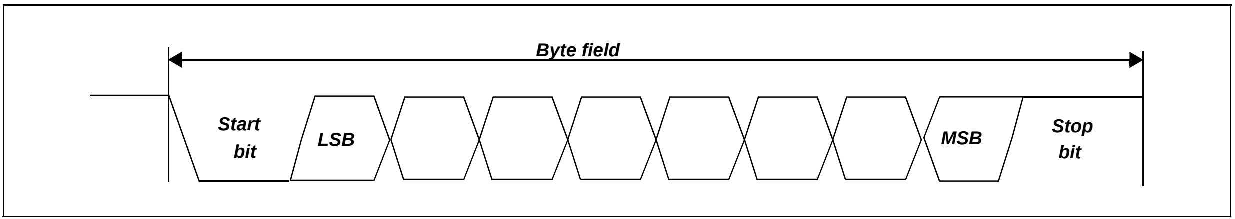
Figure 935. Structure of LIN frame



50.3.1.2 Data field

Each byte is transmitted as shown in *Figure 936*. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

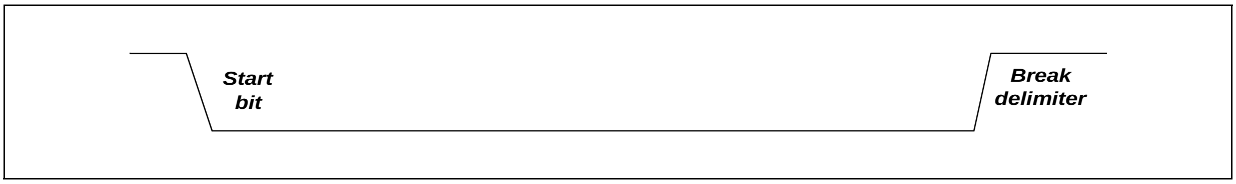
Figure 936. Structure of byte field



50.3.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in *Figure 937*. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

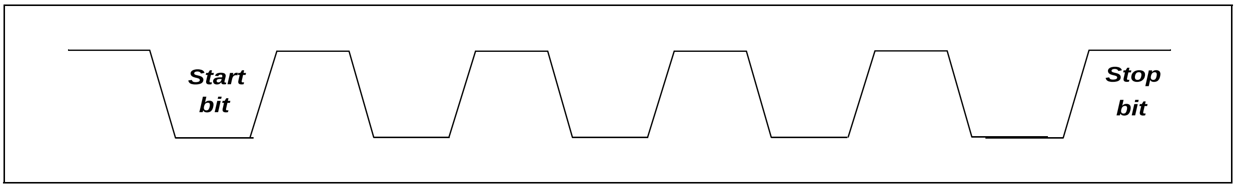
Figure 937. Break field



50.3.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

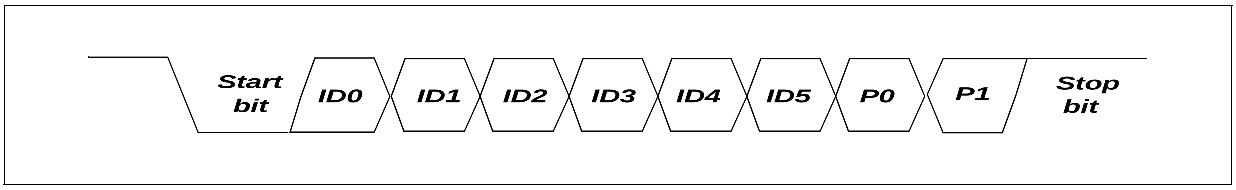
Figure 938. Sync byte field



50.3.1.5 Identifier

The Identifier field consists of two sub fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

Figure 939. Identifier field



P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

50.3.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum, and it used for communication with LIN 2.0 slaves.

50.3.2 LINFlexD features

50.3.2.1 Operating modes

The LINFlexD has three operating modes: Initialization, Normal, and Sleep modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

50.3.2.1.1 Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINCR1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller, software must:

1. Set up the baud rate registers
2. Reset the UART bit
3. Select the mode (Master or Slave)
4. Configure checksum control bits
5. Initialize the identifier list (Slave mode)

50.3.2.1.2 Normal mode (NM)

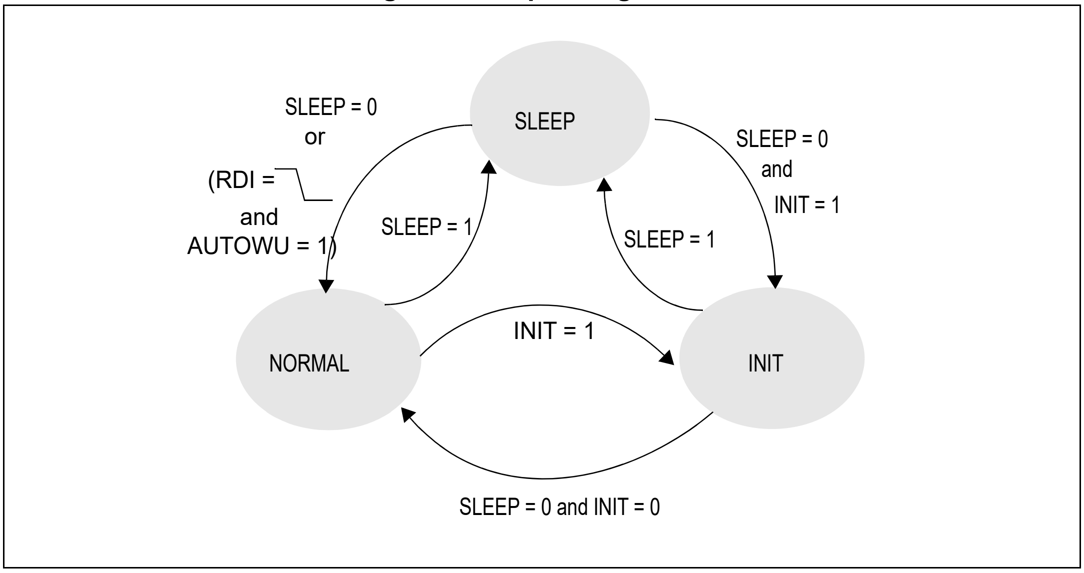
Once software has completed initialization of the LINFlexD controller, it can enter Normal mode by clearing the INIT bit.

50.3.2.1.3 Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINCR1. In this mode the LINFlexD clock is stopped. LINFlexD can be woken from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 µs on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to *Section 50.3.2.10: Wakeup management*.

Figure 940. Operating modes



50.3.2.2 Test mode

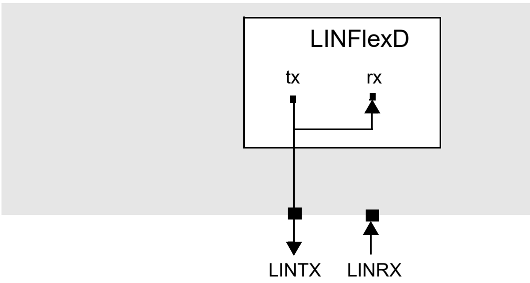
50.3.2.2.1 Loop Back mode

This mode is entered by setting the LBKM bit in LINCR1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and

Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.

Figure 941. Loop Back test mode



50.3.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

50.3.2.3.1 Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits until the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

50.3.2.3.2 Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDR[0:7]. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling). It is possible to handle frames with a response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, LINSR[DBEF] is set once the first eight bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset. Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

50.3.2.3.3 Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling). It is possible to handle frames with a Response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, the LINSR (DBFF) bit is set once the first eight bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

50.3.2.3.4 Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

50.3.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

50.3.2.4.1 Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. Read the received ID in the BIDR register
2. Fill the BDR[0:x] register
3. Program the CCS and DIR bits in the BIDR register
4. Specify the data field length DFL[5:0] bits in the BIDR register
5. Trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7]. The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

50.3.2.4.2 Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software, you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BIDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

50.3.2.5 Errors

50.3.2.5.1 Header error

A header error is an error during header reception. The types are:

- Sync Del error (SDEF)
- Sync field error (SFEF)
- Identifier Parity error (IDPEF)

1. Sync Delimiter too short

The delimiter should be 1 for at least one bit time, otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if the HEIE bit of LINIER is set.

2. Inconsistent Sync field

This error condition is monitored differently depending on whether autosynchronization is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the inconsistent Sync Field informs:

- That the deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators,

  or
- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator). Two checks are performed in parallel:

1. The first check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

   If there is an inconsistent Sync Field error, it means that:

   D > 14.0625%

   If there is no error, it means that:

   D < 14.84375%

   If 14.0625% < D < 14.84375%, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.
2. The second check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = 0)

In this case the Sync character is received as a normal character. If the received character is 0x55, then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

1. Identifier Parity error

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

50.3.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CLK cycles. Bit error is not checked during break field transmission.

Example 1

1-bit time at 20 Kbit/s = 50 µs

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) – (6 LIN_CLK cycles) = 49.925 µs

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note: If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In *this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.*

50.3.2.5.3 Framing error

This error is flagged when a dominant state is sampled on the stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if the FEIE bit is set in LINIER.

The byte that caused framing error is also shifted to the buffer but DRF or DBFF is never set in this case.

50.3.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINIER.

50.3.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of a slave, if the buffer is not released (RMB is not reset) before reception of the next Identifier and if RBLM is set, then the ID along with the data is discarded.

50.3.2.5.6 Timeout error

Figure 942. Incomplete response (for example, missing checksum)

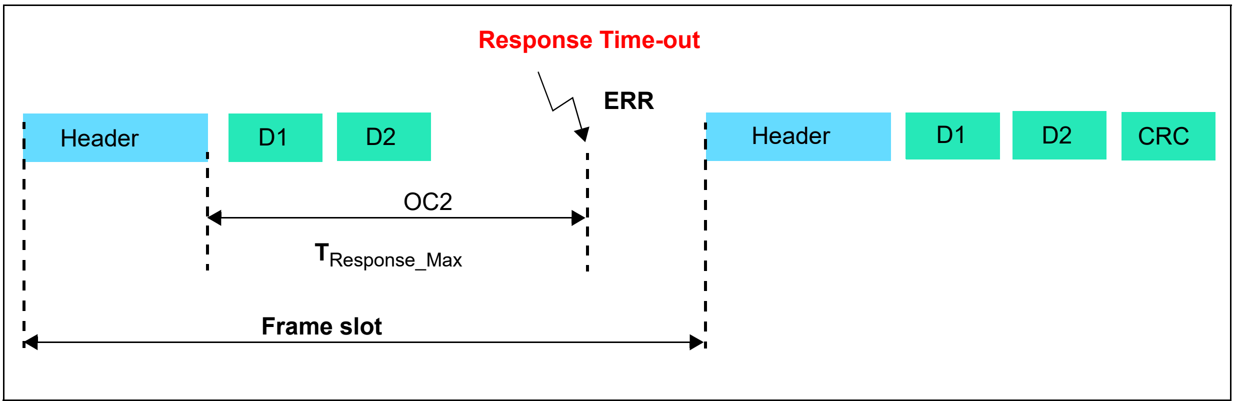
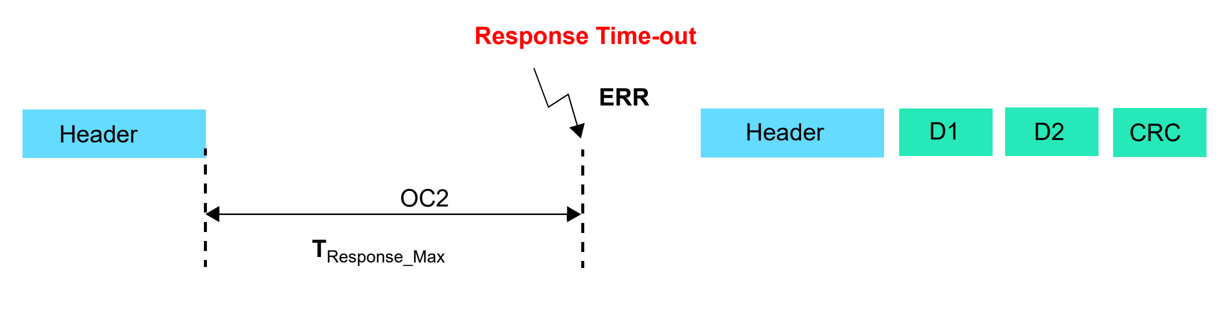


Figure 943. No response



Header timeout mechanism

- Master mode
  - ocr_2 is loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time)).
  - The change with respect to earlier implementation is that this loading takes place at the end of the ID field (and not at the beginning of the data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further ocr_2 update is required.
  - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the ocr_2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the ocr_2 value (no change compared to earlier implementation).
- Slave mode
  - **Case 1**: The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, ocr_2 can be loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time)).
  - **Case 2**: The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the previous strategy cannot be used as the DFL value is not yet available. The earlier implementation loaded the maximum value = 1.4 × (9 × 10 bit time) and then updated it according to DFL after the first data byte reception. This was not acceptable if no data byte was received at all, as timeout expired too late. The proposed implementation is the following:

    At the end of the ID, ocr_2 is loaded with 36 (maximum possible response space)

    At the end of the first_data_byte it is reloaded again according to DFL

    Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
  - no error on the bus and timing is correct (nominal header length).
  - an error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error).

  Therefore there is no meaning of header timeout in master mode, so it is disabled.
- Slave mode
  - header_nominal = 13 + 2 + 10 + 10 = 35 Tbit
  - header_max = 1.4 × Header_nominal = 49 Tbit
  - taking into account a possible 14% clock deviation, header_max seen by LINFlexD is 49 × 1.14 = 56 Tbit.
  - If the counter starts after 11 Tbit, the HTO value is 56 – 11 = 45 Tbit

  The reset value of HTO is 45 and this register can only be programmed in slave mode.

  As response space is now included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

  The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be set and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

50.3.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8^{th}, 9^{th}, and 10^{th} samples. If any one of these three samples has a value different from the other two, this error is flagged.

When OSR = 8, majority of 2^{nd}, 3^{rd}, and 4^{th} samples are taken into account to determine noise. Noise checking is disabled for all OSR less than eight.

This error is flagged when there is noise detected in the start bit (refer to *Section 50.4.2.4: LIN Error Status Register (LINESR)*).

50.3.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

50.3.2.6.1 Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2*n* and 2*n*+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2*n* or IFCR2*n*+1 (if the corresponding FACT bit is set).

50.3.2.6.2 Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2*n* is the filter and 2*n*+1 acts as a mask for it. The FACT bit for filter 2*n*+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

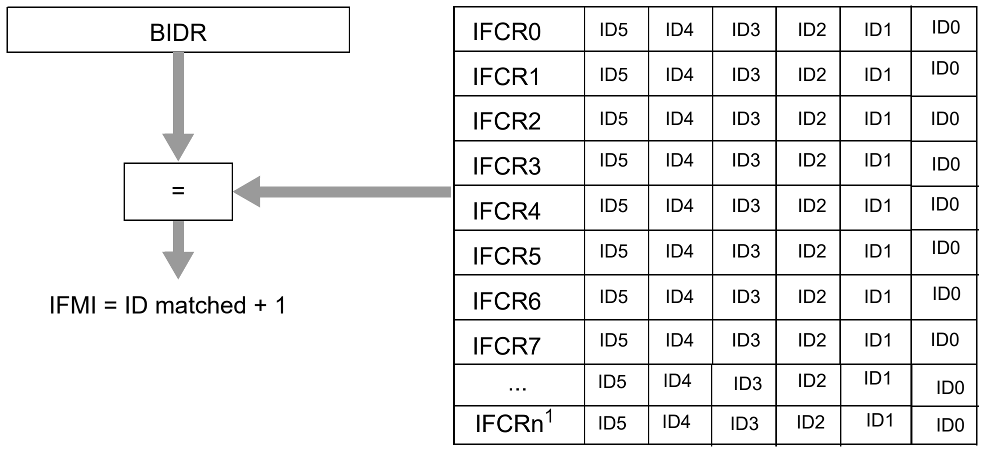
If there is a match of identifier with the *m*th filter in any mode, then *m*+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only until the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after a complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

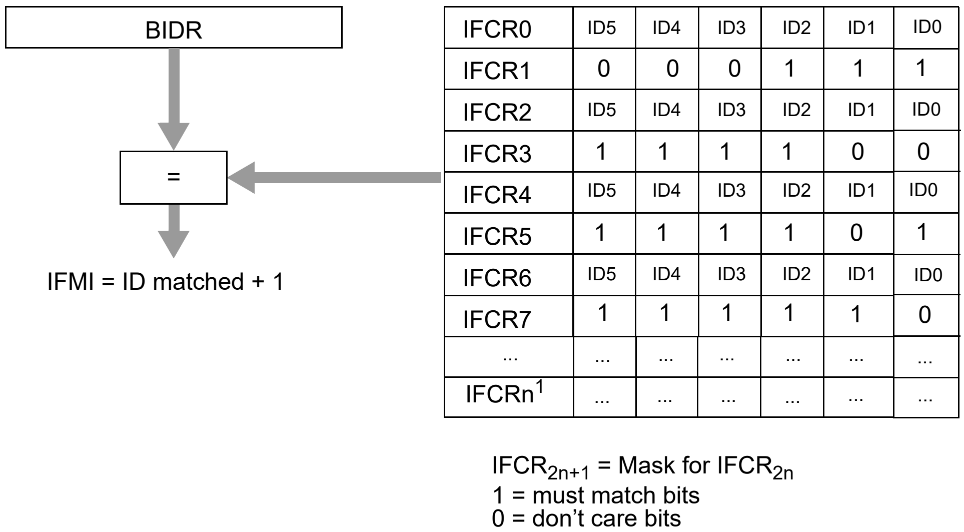
Note: If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In Mask mode, if two filters match with the identifier then the filter having the lower number prevails.

Figure 944. Identifier List mode



1. The value of n depends on the no_of_filters. Refer to the device configuration chapter to see the number of filters used in this device.

Figure 945. Identifier Mask mode



1. The value of n depends on the no_of_filters. Refer to the device configuration chapter to see the number of filters used in this device.

50.3.2.7 Start detection and delimiter detection in receiver

There is a 10-bit-shift-register sample register in the receiver that shifts signal data to the next less significant bit on each incoming sample.

Table 1009. 10-bit-shift-register sample register

Sample register bit number										Counter
9	8	7	6	5	4	3	2	1	0	
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x(1)	0	x	0	x	0	Start detection
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	Delimiter detection
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

1. x: due to sampling during transition of the bit, the bit value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. So ‘x’ refers to an indeterministic value.

50.3.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to *Table 1009* for the status of the sample register (shift register) when count = 6.
2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority of these 3 samples are equal to '0' (2 or more), then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the "verification samples" which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called "qualification samples" which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to '0' then Noise flag is set.

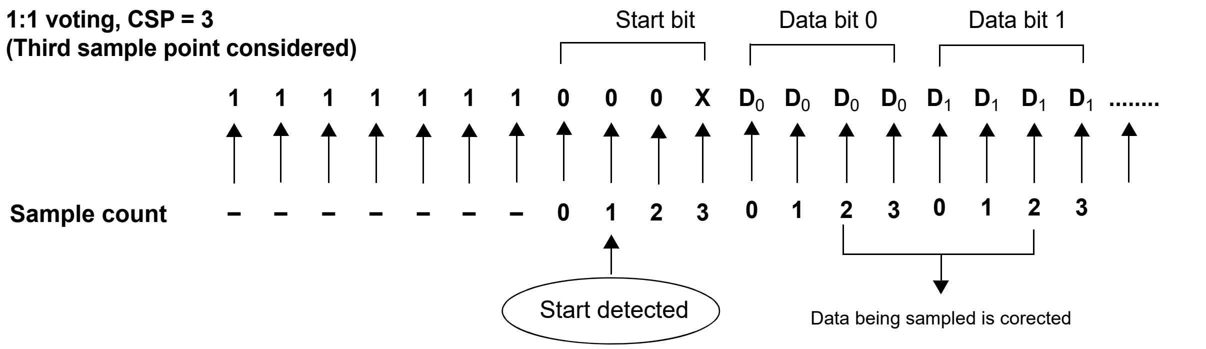
50.3.2.7.2 Delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to *Table 1009* for the status of the sample register (shift register) when count = 6.
2. At this point, if RT1(the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority of these 3 samples are equal to '1' (2 or more), then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the "verification samples" which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called "qualification samples" which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

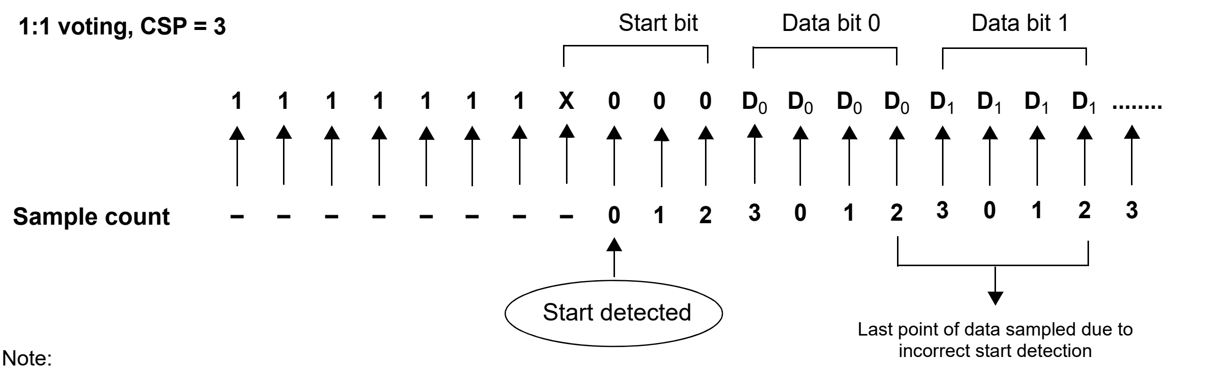
Figure 946. Start detection and sampling for over sampling rate = 4 (Case 1)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time.Therefore ‘X’ refers to a indeterministic value.

Figure 947. Start detection and sampling for over sampling rate = 4 (Case 2a)



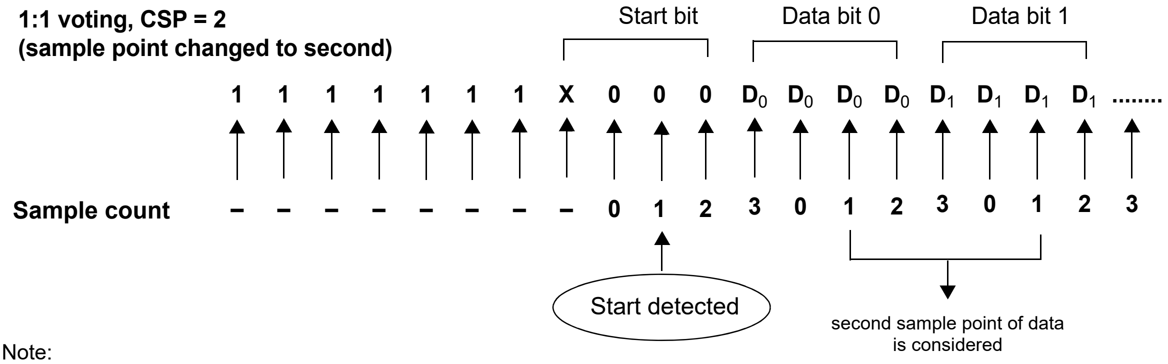
Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a indeterministic value.

The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

To sample at the correct point therefore, CSP has to be changed to two resulting in the following situation:

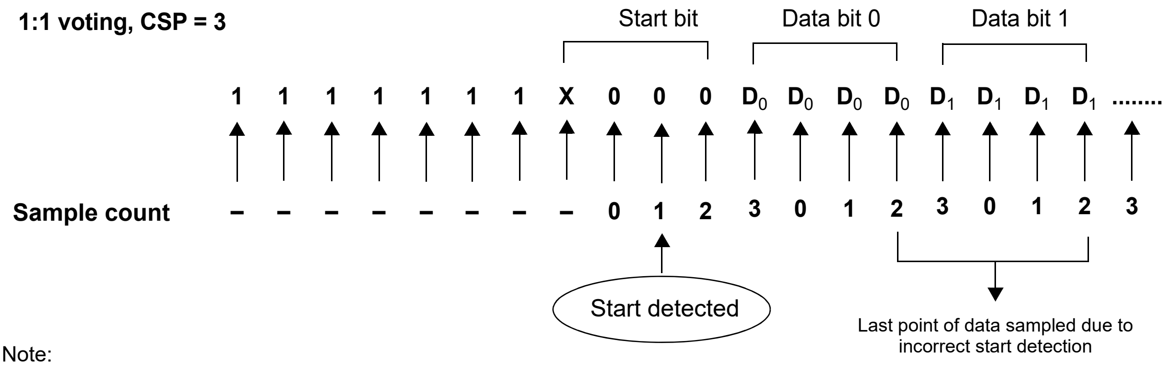
Figure 948. Start detection and sampling for over sampling rate = 4 (Case 2b)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a non deterministic value.

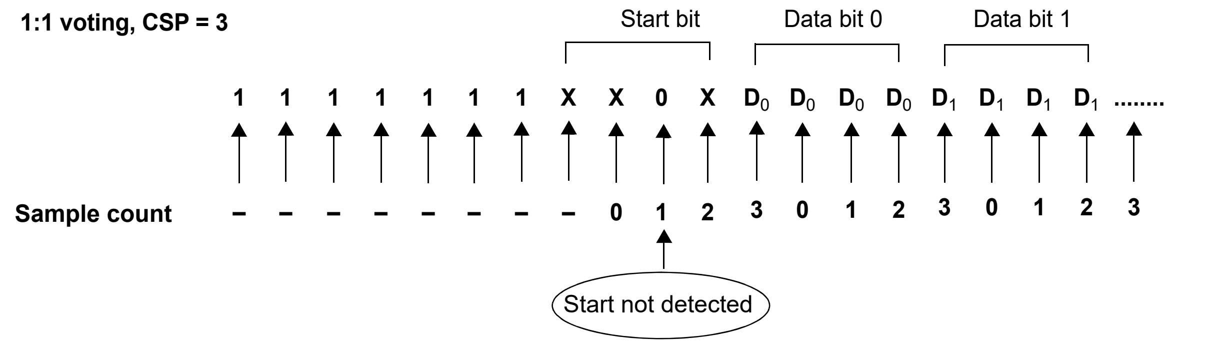
Figure 949. Start detection and sampling for over sampling rate = 4 (Case 3)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a non deterministic value.

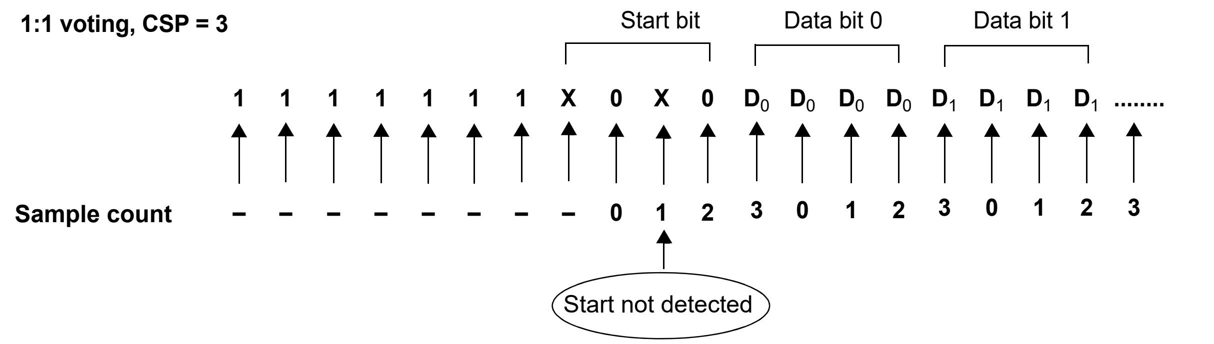
Figure 950. Start detection and sampling for over sampling rate = 4 (Case 4)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a '0' or '1' depending on the rise/fall time. Therefore 'X' refers to a non deterministic value.

Figure 951. Start detection and sampling for over sampling rate = 4 (Case 5)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a '0' or '1' depending on the rise/fall time. Therefore 'X' refers to a non deterministic value.

50.3.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate register and the LIN Fraction Baud Rate register. The baud rate registers can be programmed only during Initialization mode.

The baud rate is calculated with the following formula for both receiver and transmitter:

Equation 49 $$Tx = Rx = LIN\_CLK / (16 \times LDIV)$$

Where LIN_CLK is the frequency of the baud clock (ipg_baud_clk).

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled LINFBRR is not used and LDIV contains only the integer part of LINIBRR.

Example 2

When ROSE = 0 (For LIN and UART mode):

LDIV = 1041.67, LIN_CLK = 80 MHz

LINIBRR = 1041d = 411 h

LINFBRR = 11d = B h

Baud rate = 80 MHz/(16 x 1041,67) = 4.8 Kbit/s

Example 3

When ROSE = 1 (Only for UART mode):

LDIV = 10 d, LIN_CLK = 80 MHz

LINIBRR = 10 d, OSR = 4

Baud rate = LIN_CLK / (OSR × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s

Note: Due to design constraints the maximum baud rate achievable is 25 Mbit/s.
The relationship "(2/3)* LIN_CLK $\geq$ PBRIDGEx_CLK > 1/3*LIN_CLK" should be maintained)

50.3.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescaler value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When automatic synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

50.3.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 µs to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 µs) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

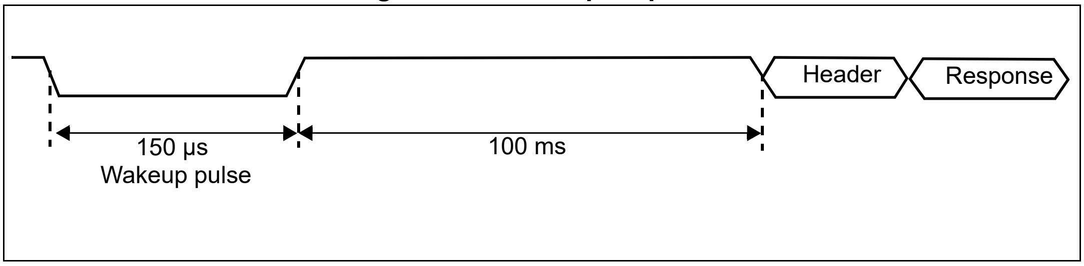
In LINFlexD, wakeup can be detected in two ways.

1. AUTOWU = 1

   On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated (if WUPIE is set). LINFlexD is now in normal mode and ready to receive frames.
2. AUTOWU = 0

   On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE is set). It is then up to the software to clear the SLEEP bit.

Figure 952. Wakeup sequence



50.3.2.11 Use case and limitations

In LIN mode, when IP is requested to enter STOP mode, the SW has to first set the INIT bit to send the receiver to IDLE. This ensures that the IP acknowledges STOP request.

50.3.3 Timer

There is an 8-bit counter which has different behavior in different modes.

In Output Compare Mode (LINTCSR[MODE]= 1): This counter is running even in Sleep and Initialization mode. The counter value which when matches the two software configurable output compare registers (LINOCR[OC1] or LINOCR[OC2]), generate output compare interrupt (LINESR[OCF]) provided TOCE bit in LINTCSR is set. Once an interrupt occurs subsequent interrupt is generated only when the application has soft reset the LIN IP through GCR [SR] bit to get reset the output compare flag (LINESR[OCF]).

In LIN mode: The software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

50.3.4 UART mode

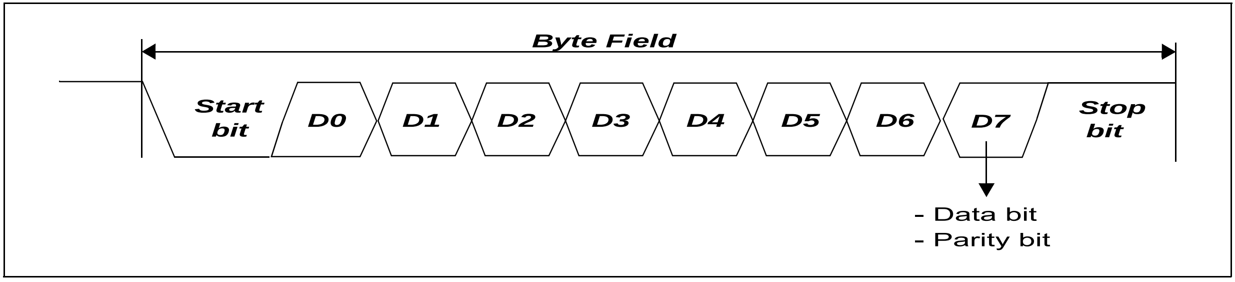
The main features in UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames,16-bit frames,17-bit frames
- Even/Odd/0/1 Parity
- User programmable over sampling rate to obtain the baud rate of up to 25 Mbit/s

50.3.4.1 8-bit data frames

The eighth bit can be a data bit or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

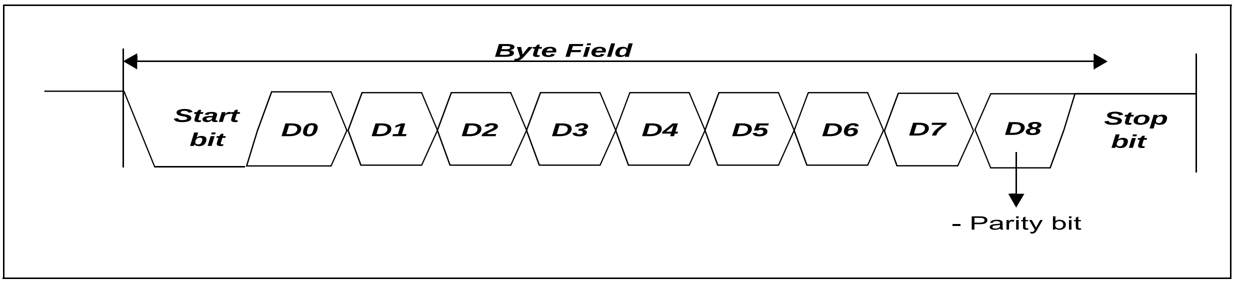
Figure 953. UART mode 8-bit data frame



50.3.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

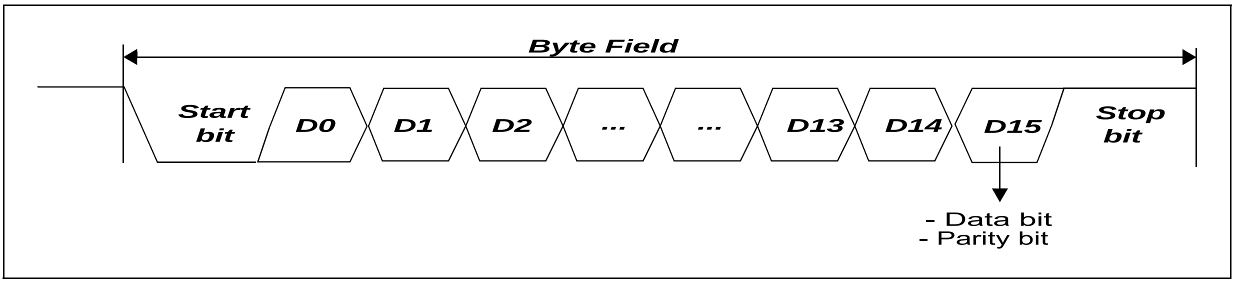
Figure 954. UART mode 9-bit data frame



50.3.4.3 16-bit data frames

The 16th bit can be a data or parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

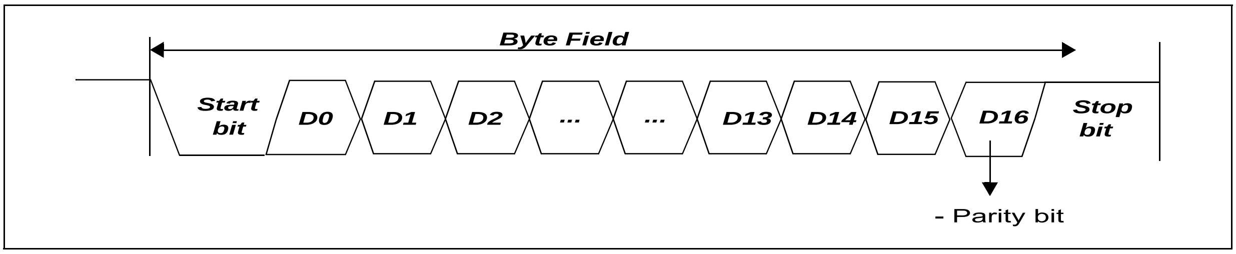
Figure 955. UART mode 16-bit data frame



50.3.4.4 17-bit frames

The 17th bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

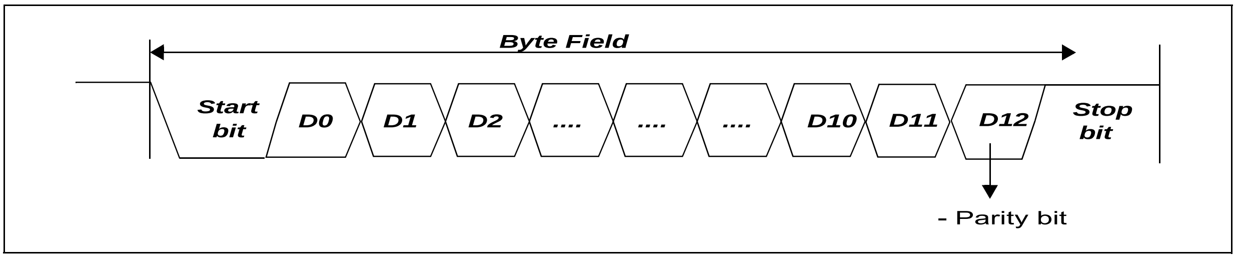
Figure 956. UART mode 17-bit data frame



50.3.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception only in FIFO mode.

Figure 957. UART mode 13-bit data frame



50.3.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in *Figure 958*.

Figure 958. Structure of 8-byte buffer

Tx0	BDR0
Tx1	BDR1
Tx2	BDR2
Tx3	BDR3
Rx0	BDR4
Rx1	BDR5
Rx2	BDR6
Rx3	BDR7

For 16-bit frames, the lower eight bits are written in BDR0 and the upper eight bits in BDR1.

The same applies for reception.

50.3.4.7 UART transmitter

In order to start transmission in UART mode, the UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the Txen bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

Note: If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Invalid writes to BDRL which generates IPS transfer error, also triggers data transmission at the same time.

When the FIFO is full, a read to DATA0 of BDRM reflects the content of DATA3 in BDRL. Proper functionality of the IP is not affected by this.

Table 1010. BDRL access in UART mode

IPS operation(1)	Register	Mode (UARTCR[TFBM])	Word length (UARTCR:WL)	IPS operation result
Write byte0	BDRL	FIFO	Byte	OK
Write byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write halfword0-1	BDRL	FIFO	Byte	IPS transfer error
Write word	BDRL	FIFO	Byte	IPS transfer error
Write byte0-1-2-3	BDRL	FIFO	Halfword	IPS transfer error
Write halfword0	BDRL	FIFO	Halfword	OK
Write halfword1	BDRL	FIFO	Halfword	IPS transfer error
Write word	BDRL	FIFO	Halfword	IPS transfer error
Read byte0-1-2-3	BDRL	FIFO	Byte/halfword	IPS transfer error
Read halfword0-1	BDRL	FIFO	Byte/halfword	IPS transfer error
Read word	BDRL	FIFO	Byte/halfword	IPS transfer error
Write byte0-1-2-3	BDRL	Buffer	Byte/halfword	OK
Write halfword0-1	BDRL	Buffer	Byte/halfword	OK
Write word	BDRL	Buffer	Byte/halfword	OK
Read byte0-1-2-3	BDRL	Buffer	Byte/halfword	OK
Read halfword0-1	BDRL	Buffer	Byte/halfword	OK
Read word	BDRL	Buffer	Byte/halfword	OK

1. Halfword0 refers to DATA1 and DATA0 of BDRL register. Halfword1 refers to DATA3 and DATA2 of BDRL register.
Halfword2 refers to DATA5 and DATA4 of BDRM register. Halfword3 refers to DATA7 and DATA6 of BDRM register.

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

50.3.4.8 UART receiver

The reception of a data byte is started as soon as the user exits Initialization mode, sets the Rxen bit, and detects the start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-halfword (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. Rxen bit needs to be set only to start the reception.The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 1011. BDRM access in UART mode

IPS operation[1]	Register [2]	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read byte4	BDRM	FIFO	Byte	OK
Read byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read halfword2-3	BDRM	FIFO	Byte	IPS transfer error
Read word	BDRM	FIFO	Byte	IPS transfer error
Read byte4-5-6-7	BDRM	FIFO	Halfword	IPS transfer error
Read halfword2	BDRM	FIFO	Halfword	OK
Read halfword3	BDRM	FIFO	Halfword	IPS transfer error
Read word	BDRM	FIFO	Halfword	IPS transfer error
Write byte4-5-6-7	BDRM	FIFO	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	FIFO	Byte/halfword	IPS transfer error
Write word	BDRM	FIFO	Byte/halfword	IPS transfer error
Read byte4-5-6-7	BDRM	Buffer	Byte/halfword	OK
Read halfword2-3	BDRM	Buffer	Byte/halfword	OK
Read word	BDRM	Buffer	Byte/halfword	OK
Write byte4-5-6-7	BDRM	Buffer	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	Buffer	Byte/halfword	IPS transfer error
Write word	BDRM	Buffer	Byte/halfword	IPS transfer error

1. Halfword0 refers to DATA1 and DATA0 of BDRL register. Halfword1 refers to DATA3 and DATA2 of BDRL register.
Halfword2 refers to DATA5 and DATA4 of BDRM register. Halfword3 refers to DATA7 and DATA6 of BDRM register.
2. Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and the data bits of the BDRL and BDRM registers.

Note: If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the *reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.*

If RDFL is programmed for a certain value but that number bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to the Idle state.

If a STOP request arrives in the middle of a reception, it is only acknowledged after all the programmed number of data bytes have been received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout; when the state machine moves to Idle state, then only a stop request is served.

If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if the FEIE bit in LINIER is set. As there is only one register bit for a framing error, this interrupt will be helpful in identifying which byte has the framing error.

If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is lost depending on the RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.

When WLS bit = 1, BDRM access depends on WL bit setting.

When WLS bit = 1, WL = 0 or 1 should not be used, since this will lead to incorrect reception of data.

In UART FIFO mode, when IP is requested to enter STOP mode, the SW has to first set the INIT bit to send the receiver to IDLE. This ensures that the IP acknowledges STOP request.

50.3.5 DMA interface

50.3.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channel where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channel where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

This release is backwards-compatible with the existing LINFlexD release if the new features are not enabled.

50.3.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in *Table 1012* and *Table 1013*.

Table 1012. LINFlexD control/status fields description

Register	Field	Level	Description
LINCR2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)
BIDR	DFL[5:0]	—	Data field length: 1 8 bytes: normal frames 1 64 bytes: extended frames
	DIR	—	Direction of the data field: 0 Reception 1 Transmission
	CCS	—	Checksum type: 0 Enhanced 1 Classic
	ID[5:0]	—	Identifier
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA 0 Buffer data is free 1 Buffer data is ready
	DBFF	High	Data buffer full (8 bytes received) and DFL > 7 Used for extended frames.
	DBEF	High	Data buffer empty (8 bytes have been transmitted) and DFL > 7 Used for extended frames.
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode)
UARTCR	FBM	—	FIFO/Buffer mode 0 Buffer mode 1 FIFO mode (mandatory in DMA mode)
UARTSR	RFE	—	Rx FIFO empty 0 Rx FIFO not empty 1 Rx FIFO empty
	TFF	—	Tx FIFO full 0 Tx FIFO not full 1 Tx FIFO full

Table 1012. LINFlexD control/status fields description (continued)

Register	Field	Level	Description
DMATXE[2**TX_CH_NUM][(1)]		High	DMA Tx channel enable (read/write) In UART or LIN master mode only channel 0 can be programmed. In LIN slave mode: # DMA TX channel = IFMI[3:0] – 1. DMATXE[x] = 0b → TX channel x disabled DMATXE[x] = 1b → TX channel x enabled
DMARXE[2**RX_CH_NUM][(1)]		High	DMA Rx channel enable (read/write) In UART or LIN master mode only the channel 0 can be programmed. In LIN slave mode: # DMA RX channel = IFMI[3:0] – 1. DMARXE[x] = 0b → RX channel x disabled DMARXE[x] = 1b → RX channel x enabled

1. ** stands for exponentiation

Table 1013. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0] & dma_enb_req_tx[0]	High	Logical AND between the two DMA enable bits (LINFlexD and eDMA)
DMA_TEN	LIN slave Tx	DMATXE[x] & dma_enb_req_tx[x] x = IFMI – 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0] & dma_enb_req_rx[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] & dma_enb_req_rx[x] x = IFMI – 1	High	

Control/status fields of the Tranfer Control Descriptors (TCDs) referred to in this document are described in *Table 1014*.

Table 1014. TCD control fields description

TCD field	Level	Description
CITER[14:0]	—	Current major iteration count
BITER[14:0]	—	Beginning major iteration count
NBYTES[31:0]	—	Inner minor byte transfer count Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset Applied to the current source address as each source read is completed.

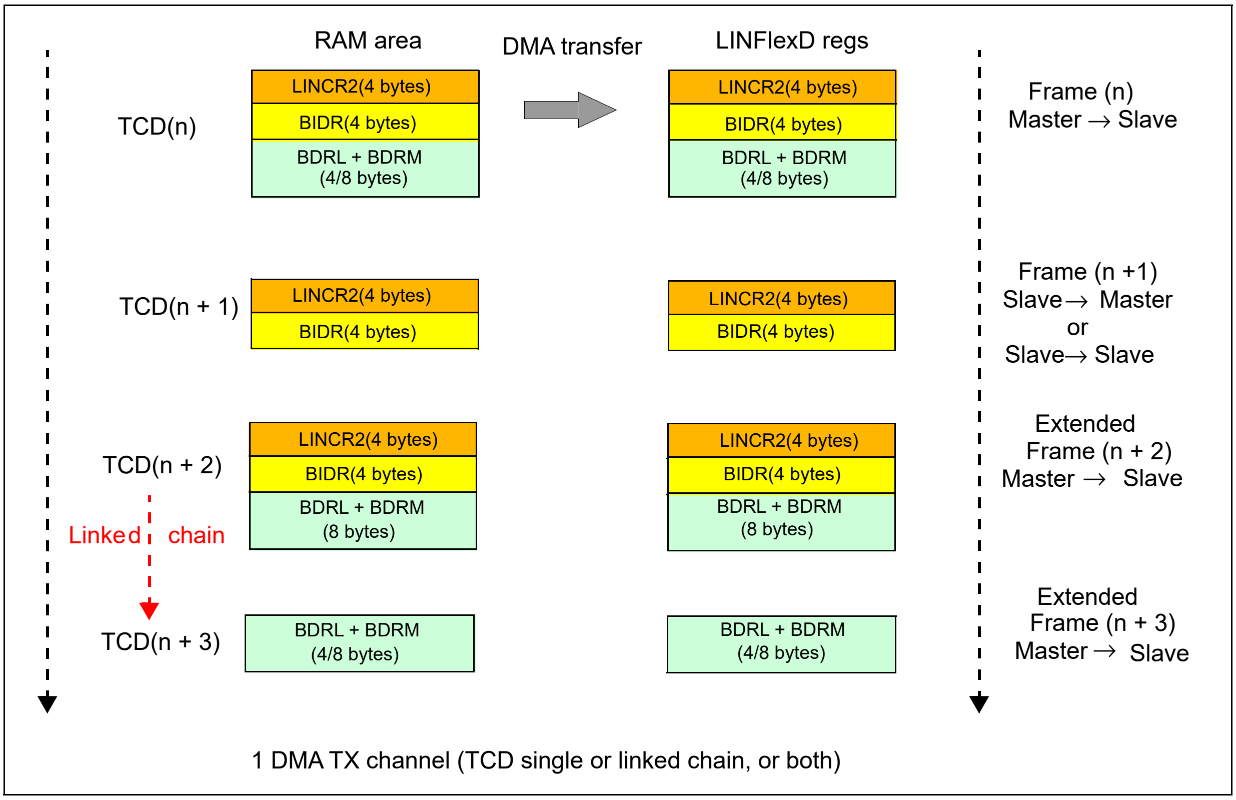
Table 1014. TCD control fields description (continued)

TCD field	Level	Description
SSIZE[2:0]	—	Source data transfer size 000 8-bit 001 16-bit 010 32-bit 011 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address
DOFF[15:0]	—	Destination address signed offset Applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 8-bit 001 16-bit 010 32-bit 011 64-bit
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request
DONE	High	Channel done (the DMA has completed the outer major loop)
D_REQ	High	Disable request If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

50.3.5.3 Master node: TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 959*.

Figure 959. Master node: TX memory map



The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

Table 1015 provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 1015. Master node: Tx mode - Register setting

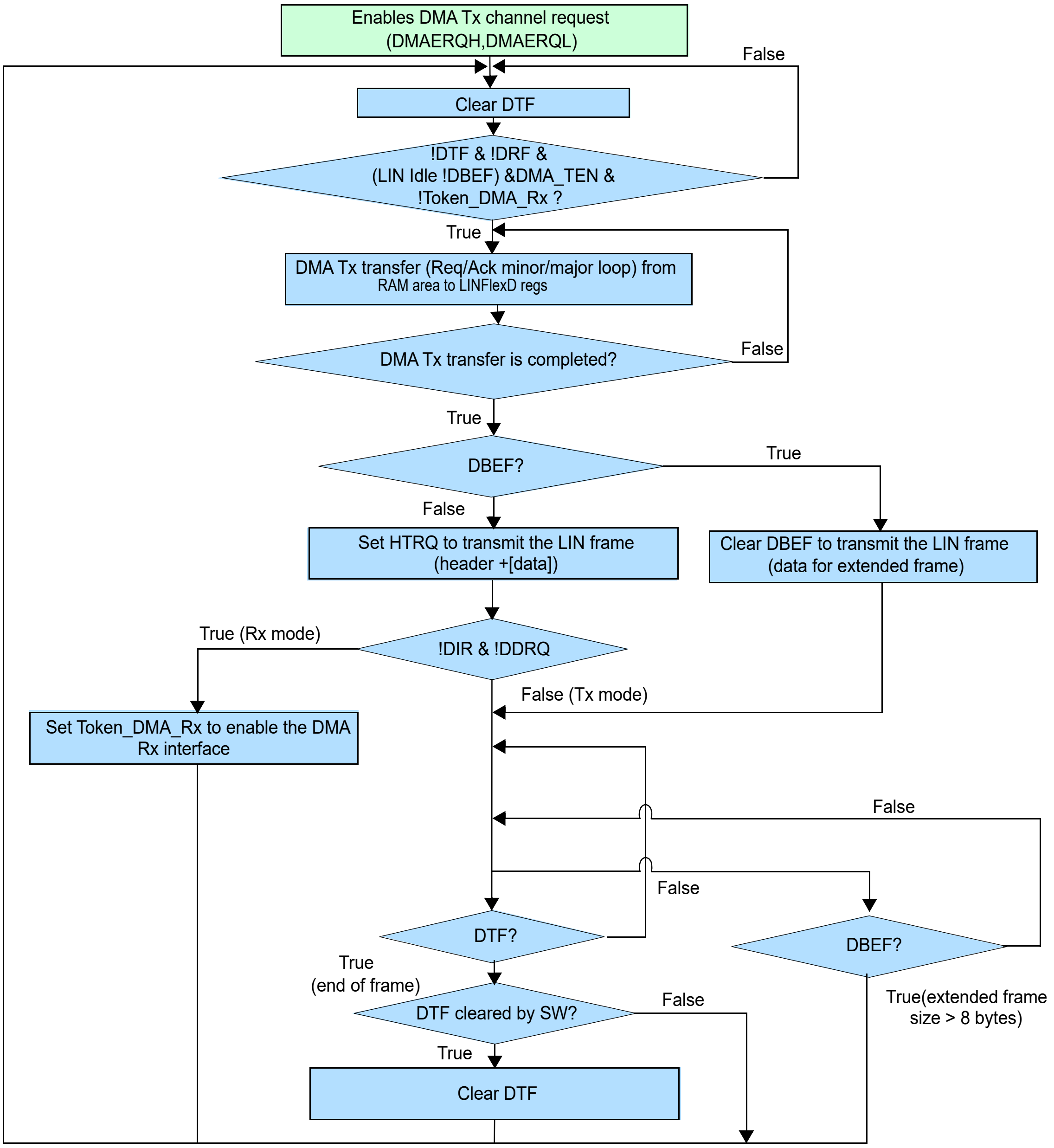
LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in *Figure 960*. DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in *Table 1016*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 1016. TCD setting: Master node -Tx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4 + 4] + 0/4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned LINCR2 + BIDR + BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	LINCR2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

Figure 960. Master node: DMA Tx FSM (concept scheme)

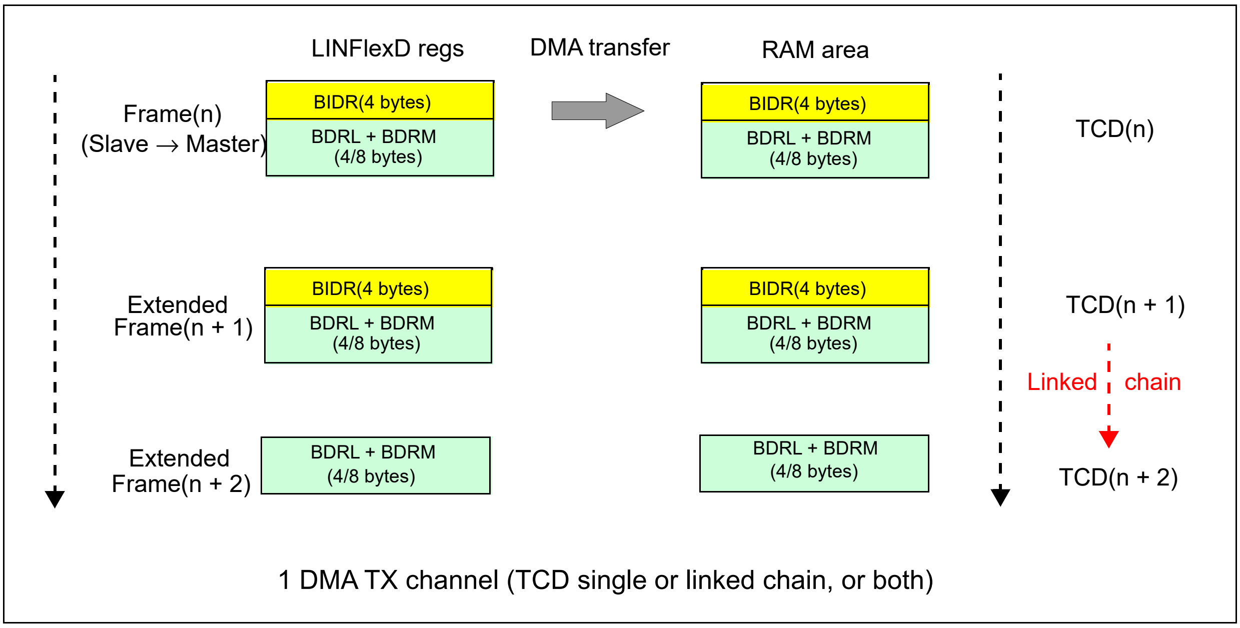


Note: DTF Flag gets auto cleared by Hardware if Software does not clear it within 3 ipg_clk cycles.

50.3.5.4 Master node: RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 961*.

Figure 961. Master node: RX memory map



The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header.

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in *Figure 962*. DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in *Table 1017*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

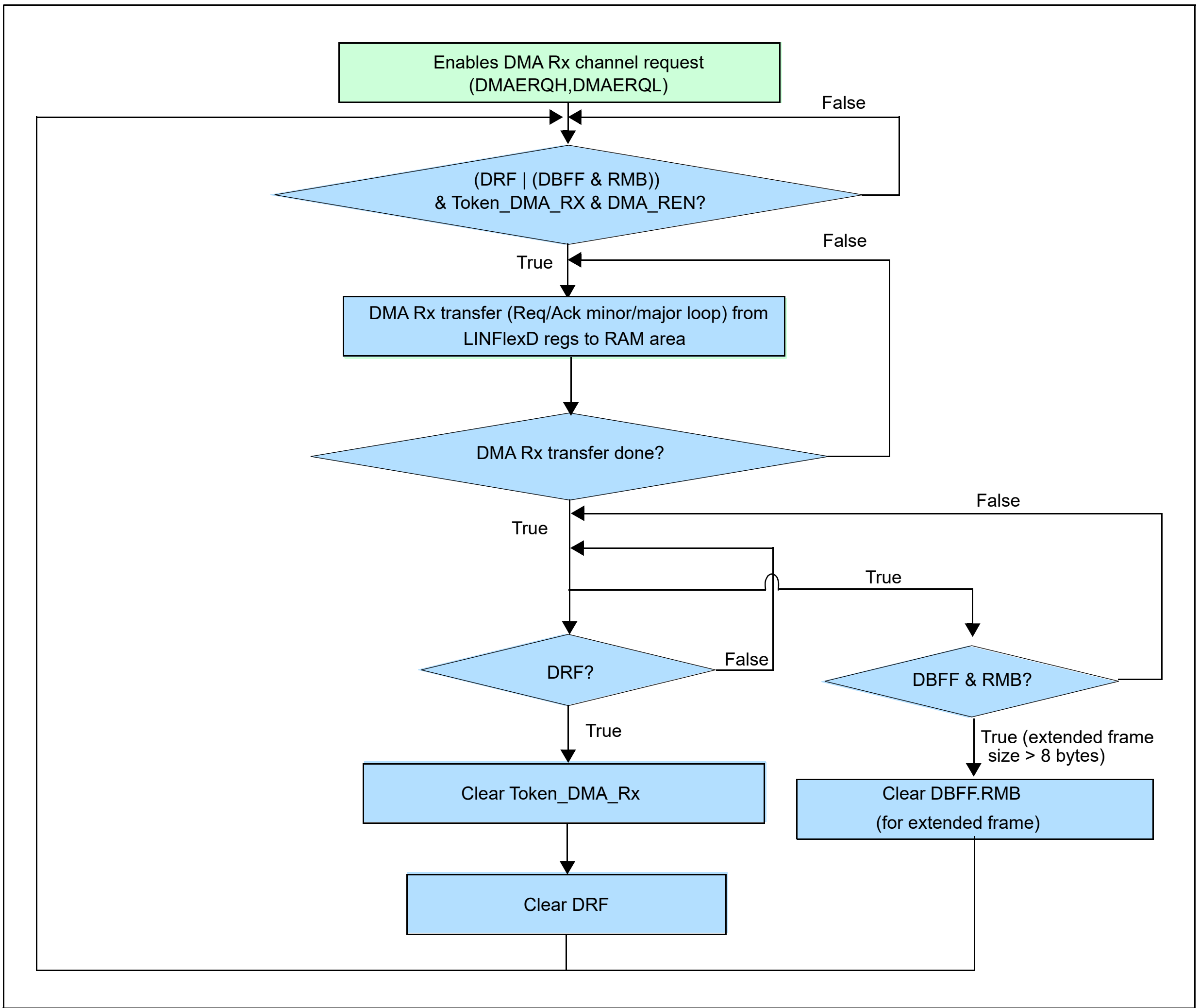
Table 1017. TCD setting: Master node - Rx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4] +4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing

Table 1017. TCD setting: Master node - Rx mode (continued)

TCD field	Value	Description
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

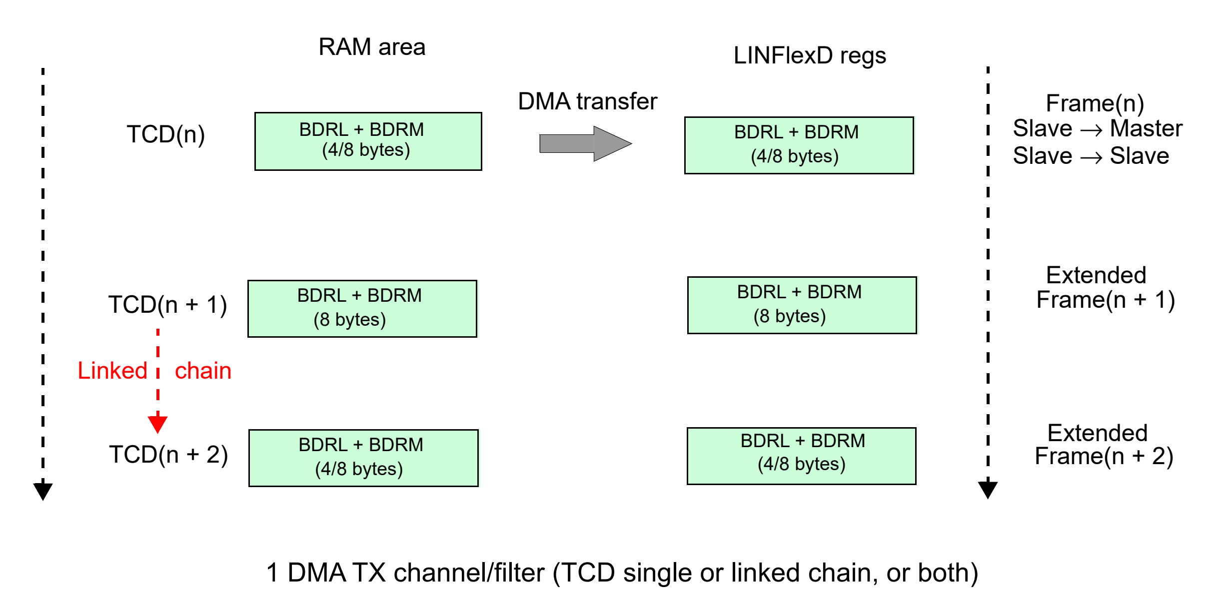
Figure 962. Master node — DMA Rx FSM (concept scheme)



50.3.5.5 Slave node: TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 963*.

Figure 963. Slave node — TX memory map



The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in *Table 1018*.

Table 1018. Slave node: Tx mode - Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Slave to Master or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Tx mode) for each DMA TX channel	– Identifier list mode Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in *Figure 964*. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 1019*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

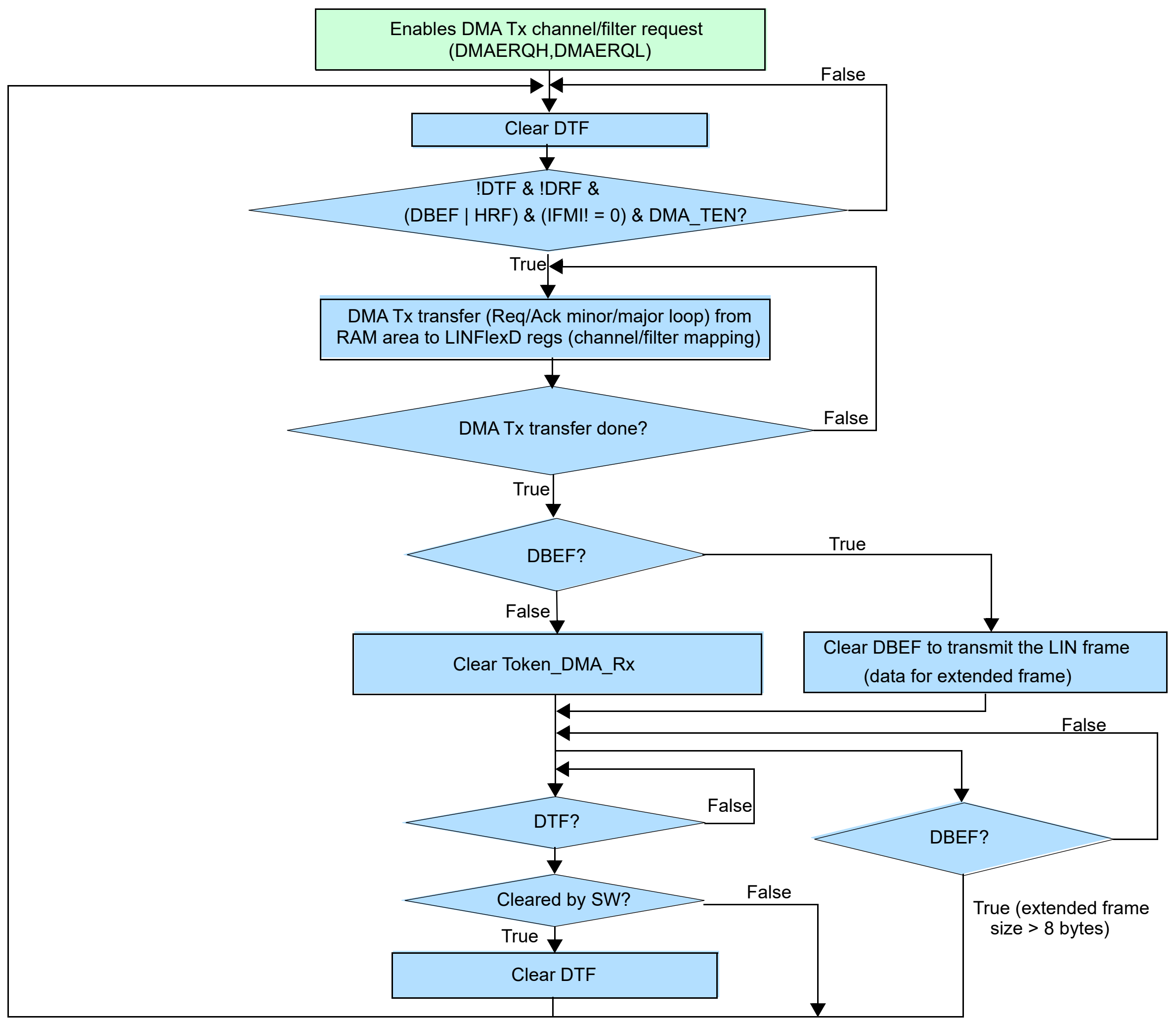
Table 1019. TCD setting: Slave node - Tx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment

Table 1019. TCD setting: Slave node - Tx mode (continued)

TCD field	Value	Description
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	-N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	-N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

Figure 964. Slave node: DMA Tx FSM (concept scheme)

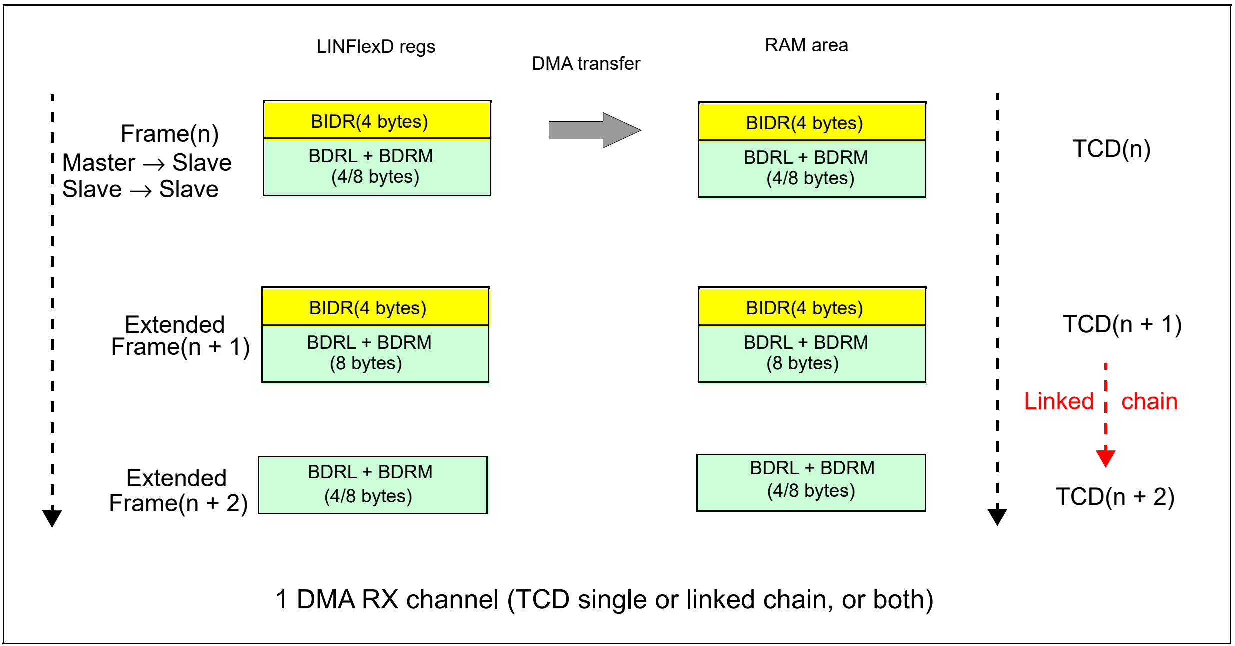


Note: DTF Flag gets auto cleared by Hardware if Software does not clear it within 3 ipg_clk cycles.

50.3.5.6 Slave node: RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 965*.

Figure 965. Slave node: RX memory map



The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in *Table 1020*.

Table 1020. Slave node: Rx mode - Register setting

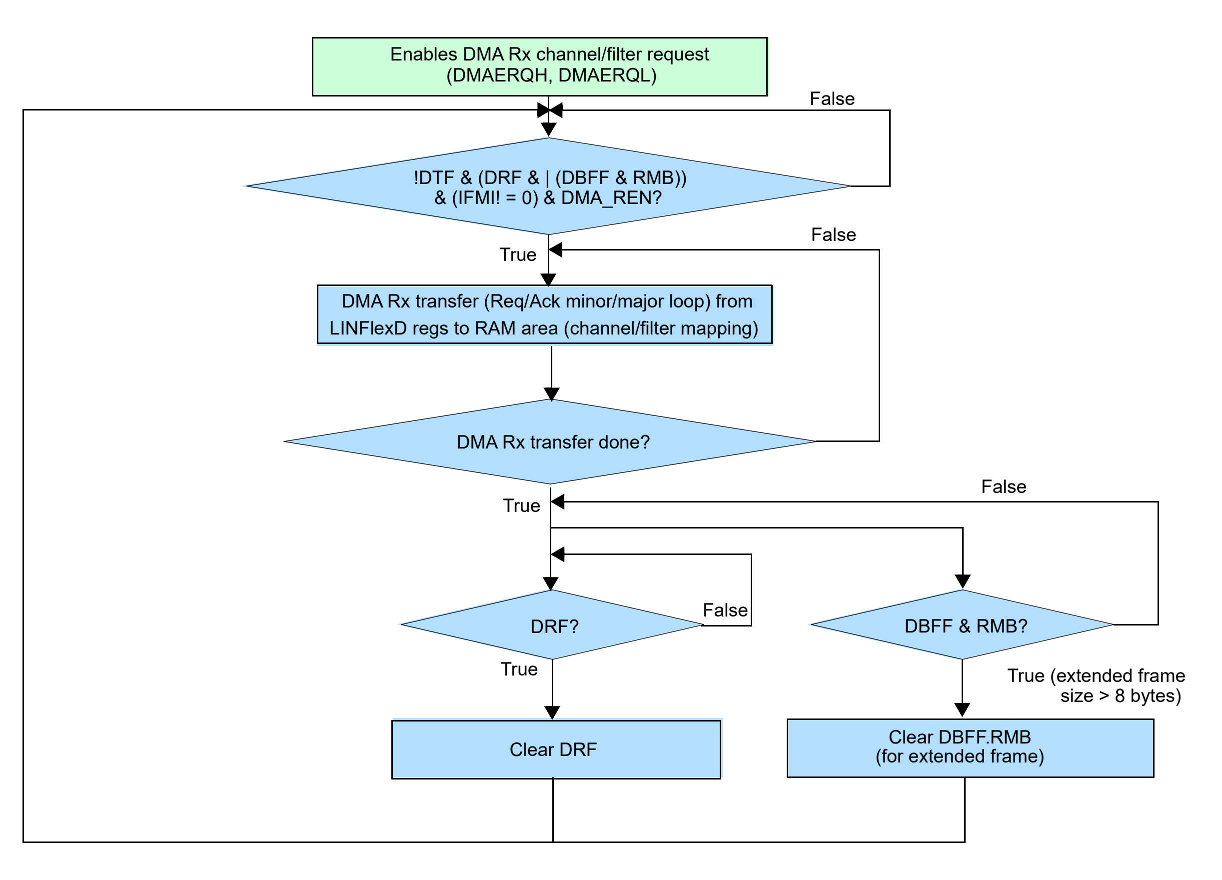
LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	– Identifier list mode Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in *Figure 966*. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 1021*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 1021. TCD setting: Slave node - Rx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	-N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	-N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

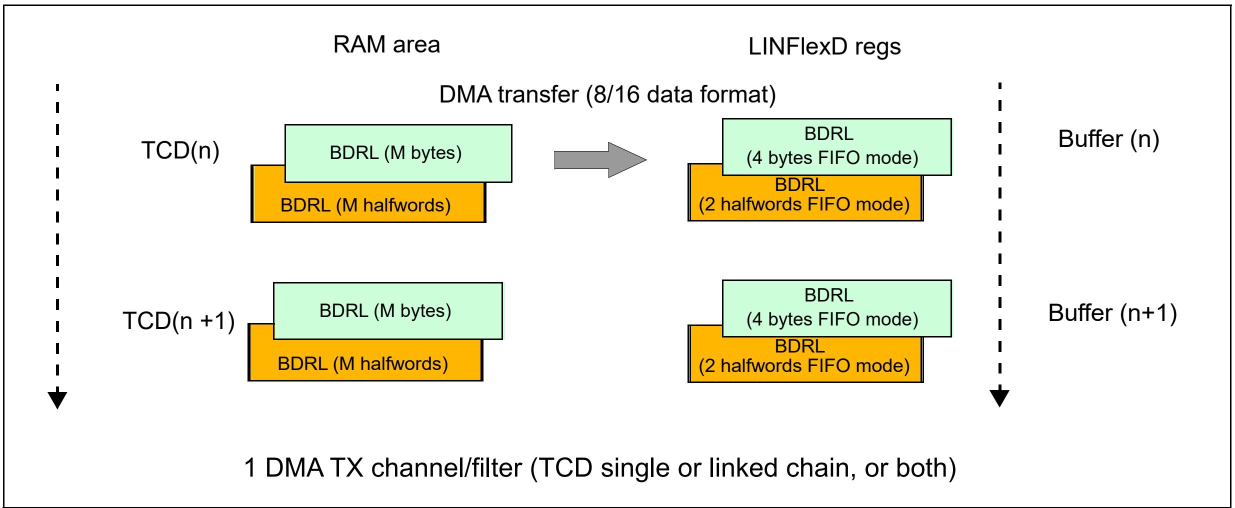
Figure 966. Slave node — DMA Rx FSM (concept scheme)



50.3.5.7 UART: TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 967*.

Figure 967. UART: TX memory map



The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8-bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

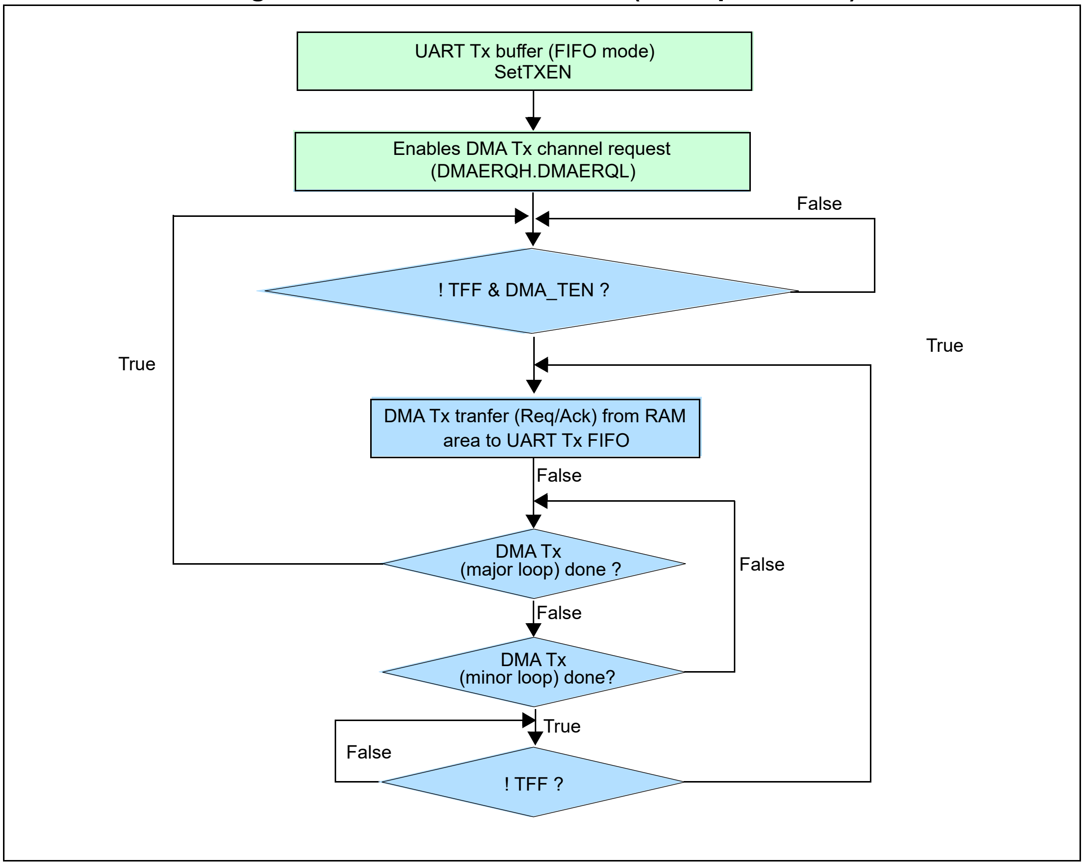
The concept FSM to control the DMA Tx interface is given in *Figure 968*. DMA Tx FSM will move to Idle state if DMATXE[0] = 0. The TCD setting (typical case) is given in *Table 1022*. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 1022. TCD setting: UART - Tx mode

TCD field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the major loop
BITER[14:0]	M		Multiple iterations for the major loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	—		RAM address
SOFF[15:0]	1	2	Byte/halfword increment
SSIZE[2:0]	0	1	Byte/halfword transfer
SLAST[31:0]	-M	-M × 2	—
DADDR[31:0][1]	—		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/halfword transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain
START	0		No software request

1. DADDR[31:0] = BDRL + 0x3 for byte transfer — DADDR [31:0] = BDRL + 0x2 for halfword transfer.

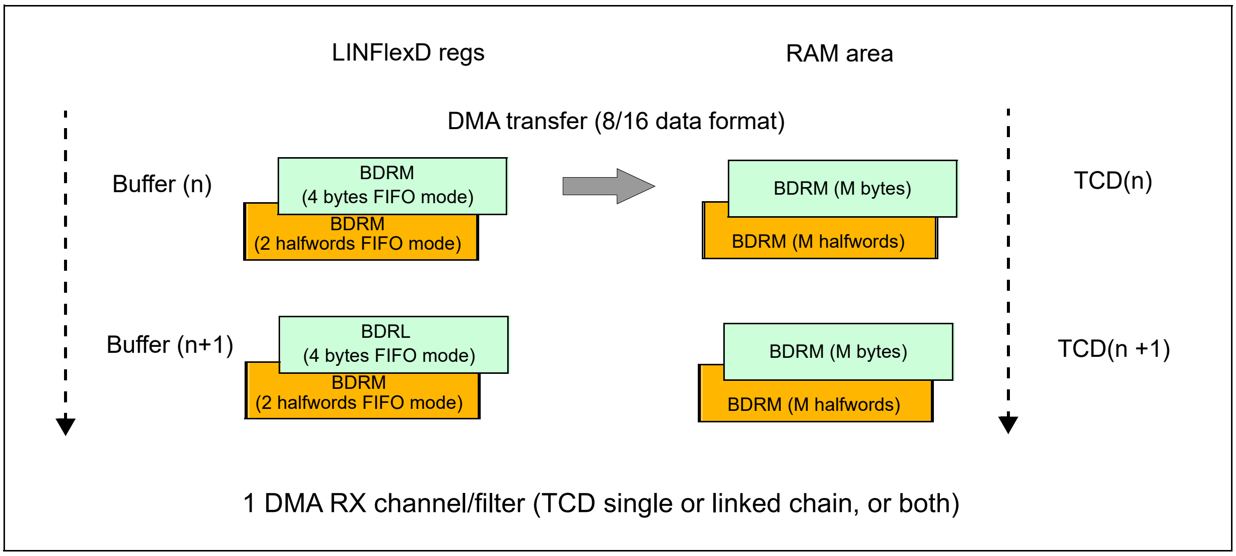
Figure 968. UART: DMA Tx FSM (concept scheme)



50.3.5.8 UART: RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Table 1023*.

Figure 969. UART: RX memory map



The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format
- 2 halfwords in 16-bit data format

This will be sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

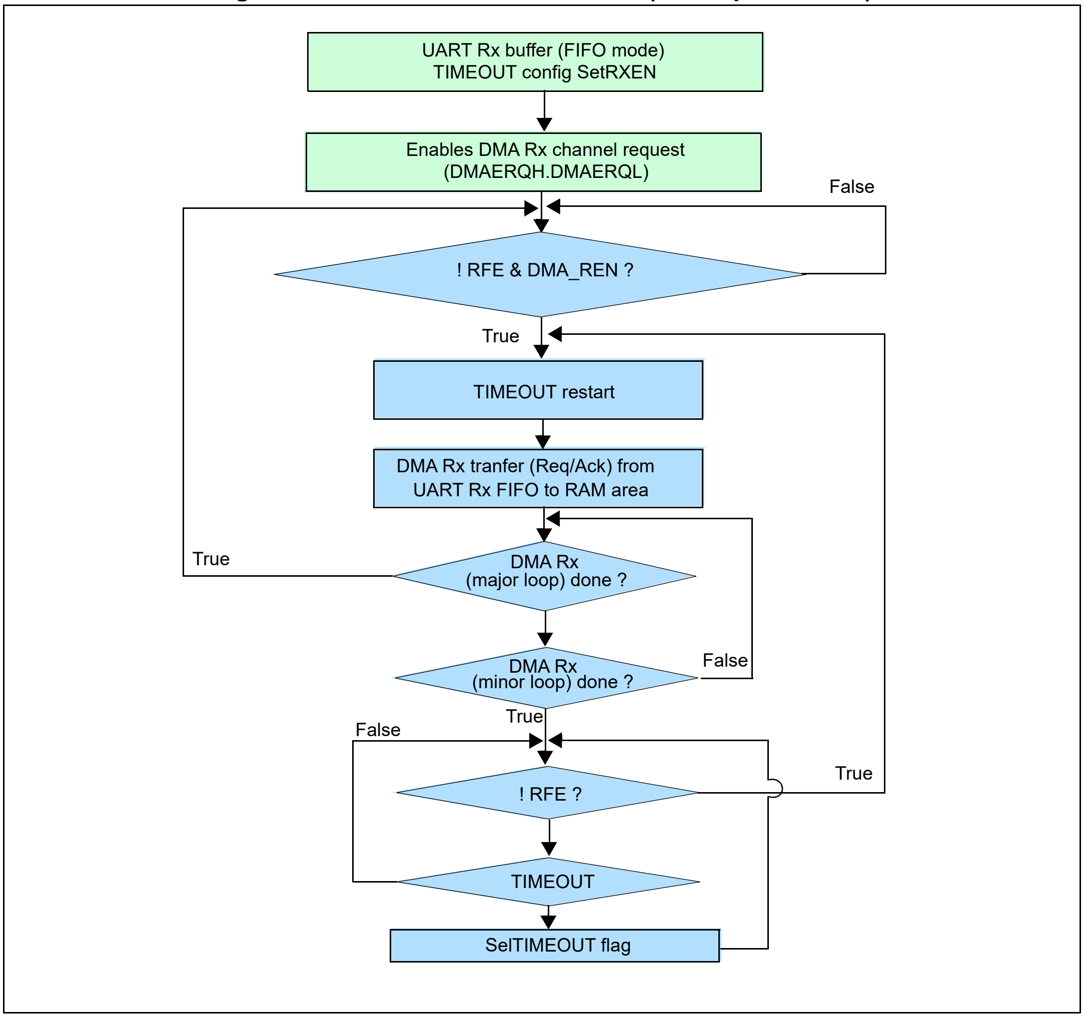
The concept FSM to control the DMA Rx interface is given in *Figure 970*. DMA Rx FSM will move to Idle state if DMARXE[0] = 0. The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/halfword as soon an entry is available in the Rx FIFO. A new software reset bit is required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 1023. TCD setting: UART - Rx mode

TCD field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the major loop
BITER[14:0]	M		Multiple iterations for the major loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0](1)	—		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/halfword transfer
SLAST[31:0]	0		—
DADDR[31:0]	—		RAM address
DOFF[15:0]	1	2	Byte/halfword increment
DSIZE[2:0]	0	1	Byte/halfword transfer
DLAST_SGA[31:0]	-M	-M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain
START	0		No software request

1. SADDR = BDRM + 0x3 for byte transfer; SADDR[31:0] = BDRM + 0x2 for halfword transfer.

Figure 970. UART — DMA Rx FSM (concept scheme)



50.3.5.9 DMA interface timing

The LINFlexD DMA interface includes the following signals:

- ipd_req
  - DMA peripheral request, asserted until the DMA transfer has been completed.
- dma_done_lw
  - DMA minor loop done. It indicates that the channel's data transfer of nbytes (the minor loop) has been completed. It is asserted for one cycle after the data phase of the channel's last AMBA AHB write access, or when an error condition is detected.
- dma_ipd_complete
  - DMA major loop finished. It is asserted for one cycle after the data phase of the last write.
- dma_ipd_ack, dma_ipd_done
  - Unused in this implementation.

All signals of the DMA interface must be registered in order to meet timing requirements of the DMA engine.

Figure 971. DMA interface (major loop — single iteration)

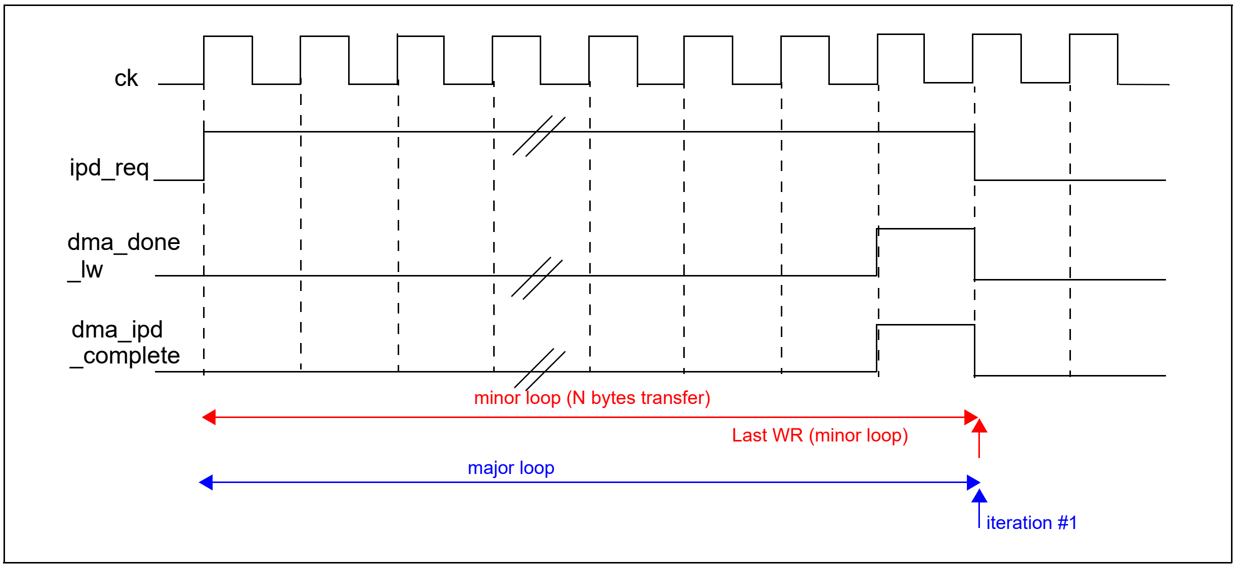
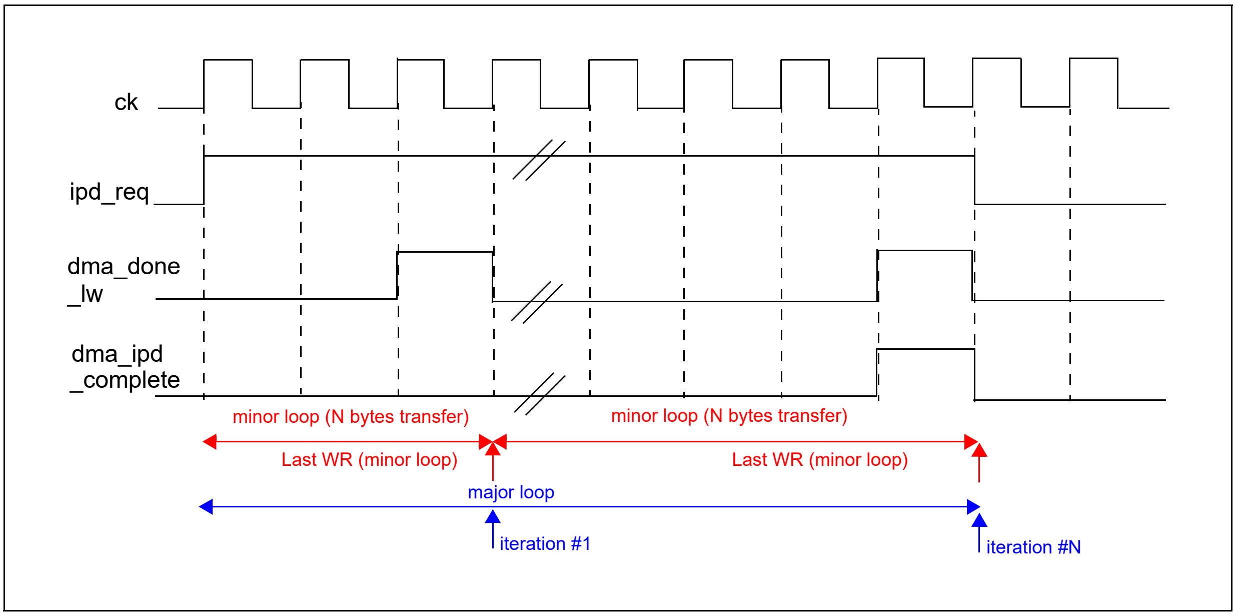


Figure 972. DMA interface (major loop — N iteration)



50.3.5.10 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index).

- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required.
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs.
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses.
- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers.
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

50.4 Memory map and register description

Note: Refer to the LINFlexD configuration section of the Device Configuration chapter for registers implementation per LINFlexD instance.

50.4.1 Memory map

Table 1024 shows the LINFlexD memory map.

Note: In Master mode, the registers IFCR0–IFCR15 are not present.

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in bus transfer error.

Table 1024. LINFlexD memory map

Address offset (hex)	Register description	Location
0x00	LIN Control register 1 (LINCR1)	Section 50.4.2.1
0x04	LIN Interrupt Enable register (LINIER)	Section 50.4.2.2
0x08	LIN Status register (LINSR)	Section 50.4.2.3
0x0C	LIN Error Status register (LINESR)	Section 50.4.2.4
0x10	UART Mode Control register (UARTCR)	Section 50.4.2.5
0x14	UART Mode Status register (UARTSR)	Section 50.4.2.6
0x18	LIN Timeout Control Status register (LINTCSR)	Section 50.4.2.7
0x1C	LIN Output Compare register (LINOCR)	Section 50.4.2.8
0x20	LIN Timeout Control register (LINTOCR)	Section 50.4.2.9

Table 1024. LINFlexD memory map (continued)

Address offset (hex)	Register description	Location
0x24	LIN Fractional Baud Rate register (LINFBRR)	*Section 50.4.2.10*
0x28	LIN Integer Baud Rate register (LINIBRR)	*Section 50.4.2.11*
0x2C	LIN Checksum Field register (LINCFR)	*Section 50.4.2.12*
0x30	LIN Control register 2 (LINCR2)	*Section 50.4.2.13*
0x34	Buffer Identifier register (BIDR)	*Section 50.4.2.14*
0x38	Buffer Data Register Least Significant (BDRL)	*Section 50.4.2.15*
0x3C	Buffer Data Register Most Significant (BDRM)	*Section 50.4.2.16*
0x40	Identifier Filter Enable register (IFER)	*Section 50.4.2.17*
0x44	Identifier Filter Match index (IFMI)	*Section 50.4.2.18*
0x48	Identifier Filter Mode register (IFMR)	*Section 50.4.2.19*
0x4C	Identifier Filter Control registers (IFCR0)	*Section 50.4.2.20*
0x50	Identifier Filter Control registers (IFCR1)	*Section 50.4.2.21*
0x54	Identifier Filter Control registers (IFCR2)	*Section 50.4.2.20*
0x58	Identifier Filter Control registers (IFCR3)	*Section 50.4.2.21*
0x5C	Identifier Filter Control registers (IFCR4)	*Section 50.4.2.20*
0x60	Identifier Filter Control registers (IFCR5)	*Section 50.4.2.21*
0x64	Identifier Filter Control registers (IFCR6)	*Section 50.4.2.20*
0x68	Identifier Filter Control registers (IFCR7)	*Section 50.4.2.21*
0x6C	Identifier Filter Control registers (IFCR8)	*Section 50.4.2.20*
0x70	Identifier Filter Control registers (IFCR9)	*Section 50.4.2.21*
0x74	Identifier Filter Control registers (IFCR10)	*Section 50.4.2.20*
0x78	Identifier Filter Control registers (IFCR11)	*Section 50.4.2.21*
0x7C	Identifier Filter Control registers (IFCR12)	*Section 50.4.2.20*
0x80	Identifier Filter Control registers (IFCR13)	*Section 50.4.2.21*
0x84	Identifier Filter Control registers (IFCR14)	*Section 50.4.2.20*
0x88	Identifier Filter Control registers (IFCR15)	*Section 50.4.2.21*
0x8C	Global Control register (GCR)	*Section 50.4.2.22*
0x90	UART Preset Timeout register (UARTPTO)	*Section 50.4.2.23*
0x94	UART Current Timeout register (UARTCTO)	*Section 50.4.2.24*
0x98	DMA Tx Enable register (DMATXE)	*Section 50.4.2.25*
0x9C	DMA Rx Enable register (DMARXE)	*Section 50.4.2.26*

50.4.2 Register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format.
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of generic slave = 0 and slave = 1.

50.4.2.1 LIN Control Register 1 (LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

Address: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD(1)	CFD(1)	LASE(1)	AUTOWU(1)	MBL(1)				BF(1)	0	LBKM(1)	MME(1)	SSBL(1)	RBLM(1)	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0/1(2)	0	0	1	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. When slave = 0, this field reads ‘1’ and cannot be programmed.

Figure 973. LIN Control Register 1 (LINCR1)

Table 1025. LINCR1 field descriptions

Field	Description
16 CCD	Calculation disable 0 Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read-only. 1 Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to send a software calculated checksum/CRC (provided CFD is reset). This bit can be written during Initialization mode only. It is read-only in Normal mode.
17 CFD	Checksum field disable 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame This bit can be configured during Initialization mode only. It is read-only in Normal mode.
18 LASE	LIN Auto Synchronization Enable 0 Auto synchronization disabled 1 Auto synchronization enabled This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0).

Table 1025. LINCR1 field descriptions (continued)

Field	Description
19 AUTOWU	Auto Wakeup 0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set This bit can be configured during Initialization mode only. This bit is utilized in UART mode also.
20:23 MBL[3:0]	Master Break Length These bits choose the length of the Sync break to be generated by the master. 0x0 10-bit break length 0x1 11-bit break length 0x2 12-bit break length 0x3 13-bit break length 0x4 14-bit break length 0x5 15-bit break length 0x6 16-bit break length 0x7 17-bit break length 0x8 18-bit break length 0x9 19-bit break length 0xA 20-bit break length 0xB 21-bit break length 0xC 22-bit break length 0xD 23-bit break length 0xE 36-bit (cooling) break length 0xF 50-bit break length These bits can be programmed in Initialization mode only. They are read-only in Normal mode.
24 BF	By-pass filter 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter This bit can be programmed during Initialization mode only. Note: If generic slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed.
25	Reserved
26 LBKM	Loop Back mode 0 Loop Back Mode disabled 1 Loop Back mode enabled Refer to *Section 50.3.2.2*. Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also.
27 MME	Master mode enable 0 Slave Mode 1 Master Mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. If generic *slave* = 0, then this bit will read a ‘1’ always, and cannot be programmed.

Table 1025. LINCR1 field descriptions (continued)

Field	Description
28 SSBL	Slave Mode Sync Break Length 0 11-bit break length 1 10-bit break length This bit can be programmed in Initialization mode only. It is read-only in Normal mode.
29 RBLM	Receiver Buffer Locked mode 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun. Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software. This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also.
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also.

50.4.2.2 LIN Interrupt enable register (LINIER)

Address: 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0	0	FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIE_TOIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 974. LIN Interrupt enable register (LINIER)

Table 1026. LINIER field descriptions

Field	Description
16 SZIE	Stuck at Zero Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set.
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR.
20 HEIE	Header Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. **Note:** If generic slave = 0, then this bit will always read a 0 and cannot be programmed.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR.
25 LSIE	LIN state Interrupt enable 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag.
26 WUIE	Wakeup interrupt enable 0 No interrupt 1 Interrupt enabled If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated.

Table 1026. LINIER field descriptions (continued)

Field	Description
27 DBFIE	Data Buffer Full Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the DBFF bit in LINSR is also set.
28 DBEIE_TOIE	Data Buffer Empty Interrupt enable/Timeout Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and LINSR[DBEF] status bit is set (in LIN mode) or if this bit is set and UARTSR[TO] status bit is set (in UART mode).
29 DRIE	Data Reception complete Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set.
30 DTIE	Data Transmitted Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set.
31 HRIE	Header Received Interrupt 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

50.4.2.3 LIN Status Register (LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	AUTOSYNC_COMP	RDC		
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS				0	0	RMB	DRBNE	RXBUSY	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
W							w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 975. LIN Status Register (LINSR)

Table 1027. LINSR field descriptions

Field	Description
12 AUTOSYNC_COMP	Autosynchronization Completed This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers should be read in autosynchronization mode.
13:15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table 1027. LINSR field descriptions (continued)

Field	Description
16:19 LINS	LIN state 0x0 Sleep mode LINFlexD is in Sleep mode to save power consumption. 0x1 Init 0x2 Idle This mode is entered when: SLEEP bit and INIT bit are reset by software, Wakeup pulse has been received on RX pin (AUTOWU set), Previous frame transmission/reception has been completed/aborted. 0x3 Sync Break In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle. 0x4 Sync Del In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 0x5 Sync Field In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 0x6 Identifier Field In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 0x7 Header Reception/Transmission In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 0x8 Data Reception/Data Transmission In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 0x9 Checksum Data transmission/reception completed, checksum transmission/reception ongoing. In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.
24 RXBUSY	Receiver Busy flag 0 Receiver is idle 1 Reception ongoing In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit.

Table 1027. LINSR field descriptions (continued)

Field	Description
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **Note:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin and when: The slave is in Sleep mode The master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27 DBFF	Data Buffer full flag This bit is set by hardware when the buffer is full (8 bytes received) and DFL > 7. It should be cleared by software. This bit is also reset by hardware in Initialization mode.
28 DBEF	Data Buffer empty flag This bit is set by hardware when the buffer is empty (8 bytes have been transmitted) and DFL > 7. It should be cleared by software after data buffer is refilled by software. Transmission of next eight bytes will not start unless this bit is reset by software. This bit is also reset by hardware in Initialization mode.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **Note:** In case framing error or checksum error occurs then this flag is not set.
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **Note:** For LIN mode, in case a bit error occurs (and IOBE is reset) then this flag is not set.
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: All filters are inactive and Bypass filter (BF) bit is set No match in any filter and Bypass filter (BF) bit is set TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

50.4.2.4 LIN Error Status Register (LINESR)

Address: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	BEF	CEF	SFEF	SDEF	IDPEF	FEF	BOF	0	0	0	0	0	0	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c							w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 976. LIN error status register (LINESR)

Refer to *Section 50.3.2.5: Errors* for detailed description of all errors.

Table 1028. LINESR field descriptions

Field	Description
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of OCR1 or OCR2 This bit should be cleared by software. If this bit is set, MODE bit in LINTCSR is cleared, and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINCR1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

Table 1028. LINESR field descriptions (continued)

Field	Description
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. **Note:** Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set. If generic slave = 0, then this bit will always read a 0, and cannot be programmed. For generic slave = 1,this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

50.4.2.5 UART Mode Control Register (UARTCR)

Address: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MIS(1)	CSP(1)			OSR(1)				ROSE(1)	NEF(1)(2)			DTU_PCETX(1)	SBUR(1)		WLS(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TDFL_TFC(2)			RDFL_RFC(3)			RFBM(1)	TFBM(1)	WL1(1)	PC1(1)	RXEN	TXEN	PC0(1)	PCE(1)	WL0(1)	UART(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. When TFBM = 0, the field is read/write; when TFBM = 1, the field is read-only.
3. When RFBM = 0, the field is read/write; when RFBM = 1, the field is read-only.

Figure 977. UART Mode Control Register (UARTCR)

Table 1029. UARTCR field descriptions

Field	Description
0 MIS	Monitor Idle State 1 UARTCTO monitors the idle state of the reception line. 0 UARTCTO monitors the number of bits to be received. The default value of this bit is 0.
1:3 CSP	Configurable Sample Point (i) These bits will decide the sample point during reduced over sampling (ROSE=1). CSP can take the following range of values for a certain over sampling rate: – OSR 4: CSP 2, 3 – OSR 5: CSP 2, 3, 4 – OSR 6: CSP 3, 4, 5 – OSR 8: CSP NA For any values of CSP other than the ones mentioned in the above table, data sampled is erroneous.
4:7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled (ROSE=1). For proper functionality, OSR can only take values 4, 5, 6, and 8. For any other values start bit of data is not detected by receiver and thus the reception will never start.
8 ROSE	Reduced Over Sampling Enable 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate. The default value of this bit is 0.
9:11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received.
12 DTU_PCETX	Disable Timeout in UART mode 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received This bit can be programmed in initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received (num_of_frames). When num_of_frames equals NEF, Disable Timeout is set and num_of_frames gets reset to zero, restarting the timer and enabling the timeout again.
13:14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 Reserved This bit can be programmed in the initialization mode only, when the UART bit is set.

Table 1029. UARTCR field descriptions (continued)

Field	Description
15 WLS	Special Word Length in UART mode 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support. This bit can be programmed in initialization mode only when the UART bit is set. Parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode.
16:18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter TDFL_TFC defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL_TFC is a read/write field. TDFL_TFC[2] is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations TDFL_TFC = x01 or TDFL_TFC = x11 are allowed. x00 1-byte x01 2-bytes x10 3-bytes x11 4-bytes TDFL_TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TDFL_TFCx is a read-only field available for debug purposes. 000 empty 001 1-byte 010 2-bytes 011 3-bytes 100 4-bytes Others: Reserved **Note:** TDFL_TFC can be programmed and are significative only when the UART bit is set.
19:21 RDFL_RFC	Reception Data Field Length/RX FIFO Counter RDFL_RFC defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL_RFC is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations RDFL_RFC = x01 or RDFL_RFC = x11 are allowed. x00 1-byte x01 2-bytes x10 3-bytes x11 4-bytes RDFL_RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RDFL_RFCx is a read-only field available for debug purposes. 000 Empty 001 1-byte 010 2-bytes/1.5 byte in case of WLS bit setting 011 3-bytes 100 4-bytes/2×1.5 byte in case of WLS bit setting Others: Reserved RDFL_RFC can be programmed and are significative only when the UART bit is set. **Note:** In buffer mode, RDFL_RFC should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint.

Table 1029. UARTCR field descriptions (continued)

Field	Description
22 RFBM	RFBM Rx FIFO/Buffer mode 0 Rx Buffer mode enabled 1 Rx FIFO mode enabled (mandatory in DMA Rx mode) This bit can be programmed in Initialization mode, only when the UART bit is set.
23 TFBM	Tx FIFO/Buffer mode 0 Tx Buffer mode enabled 1 Tx FIFO mode enabled (mandatory in DMA Tx mode) This bit can be programmed in initialization mode, only when the UART bit is set.
24 WL1	Word Length in UART mode 00 7 bits data + parity 01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1 10 15 bits data + parity 11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1 This bit can be programmed in Initialization mode only, when the UART bit is set. **Note:** The two-digit binary values are a combination of the fields “WL1” and “WL0” to describe the functionality of word length in UART mode.
25 PC1	Parity Control 00 Parity sent is Even 01 Parity sent is Odd 10 A logical 0 is always transmitted/checked as parity bit 11 A logical 1 is always transmitted/checked as parity bit This bit can be programmed in Initialization mode, only when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “PC1” and “PC0” to describe the functionality of word length in UART mode.
26 RXEN	Receiver Enable 0 Receiver disabled 1 Receiver enabled This bit can be programmed only when the UART bit is set.
27 TXEN	Transmitter Enable 0 Transmitter disabled 1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed This bit can be programmed only when UART bit is set.
28 PC0	Parity Control 00 Parity sent is Even 01 Parity sent is Odd 10 A logical 0 is always transmitted/checked as parity bit 11 A logical 1 is always transmitted/checked as parity bit This bit can be programmed in Initialization mode, only when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “PC1” and “PC0” to describe the functionality of word length in UART mode.
29 PCE	Parity Control Enable 0 Parity transmission, reception, and checking disabled 1 Parity transmission, reception, and checking enabled This bit can be programmed in Initialization mode only, when the UART bit is set.

Table 1029. UARTCR field descriptions (continued)

Field	Description
30 WL0	Word Length in UART mode 00 7 bits data + parity 01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1 10 15 bits data + parity 11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1 This bit can be programmed in Initialization mode only, when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “WL1” and “WL0” to describe the functionality of word length in UART mode.
31 UART	UART Mode 0 LIN mode 1 UART mode This bit can be programmed in Initialization mode only.

50.4.2.6 UART Mode Status Register (UARTSR)

Address: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE				RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF_RFE(1)	DTF_TFF(2)	NF
W	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c		w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. When RFBM = 0, the field can be read/cleared by software. Writing 1 clears contents; when RFBM = 1, the field is read-only.
2. When TFBM = 0, the field can be read/cleared by software. Writing 1 clears contents; when TFBM = 1, the field is read-only.

Figure 978. UART Mode Status Register (UARTSR)

Table 1030. UARTSR field descriptions

Field	Description
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCR An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. For Baud rate above 1 Mbit/s, this flag is not usable.
18:21 PE	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. PE[i] (where i = 0,1,2 or 3) can be equal to: 0 No parity error 1 Parity error in the corresponding received byte This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. **Note:** Parity is checked only after a complete frame is received. When WL bits = 10 or 11, either PE[0] or PE[2] is only set. – When PE[0] is set, it indicates Parity error in either first or second byte received. – When PE[2] is set, it indicates Parity error in either third or fourth byte received.
22 RMB	Release Message Buffer 0 Buffer data is free 1 Buffer data ready to be read by software This bit should be cleared by software. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.

Table 1030. UARTSR field descriptions (continued)

Field	Description
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software.The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode. An interrupt will be generated when LINIER.DBEIE/TOIE bit is set on the Error interrupt line in UART mode.
29 DRF_RFE	Data Reception Completed Flag/Rx FIFO Empty Flag DRF_RFE is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF_RFE should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF_RFE bit is set only when the configured number of stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF_RFE is set irrespective of framing error, parity error or overrun error. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. DRF_RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. DRF_RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTF_TFF	Data Transmission Completed Flag/TX FIFO Full Flag DTF_TFF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF_TFF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. DTF_TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. DTF_TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

50.4.2.7 LIN Time-Out Control Status Register (LINTCSR)

This register contains control and status bits for the timeout feature.

Address: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MODE(1)	IOT(1)	TOCE	CNT							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 979. LIN Time-Out Control Status Register (LINTCSR)

Table 1031. LINTCSR field descriptions

Field	Description
21 MODE	Time-out counter mode 0 LIN mode 1 Output compare mode This bit can be configured only during initialization.
22 IOT	Idle on timeout 0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event This feature is applicable only when MODE bit in LINTCSR is cleared.
23 TOCE	Time-out counter enable 0 Time-out counter disable. OCF flag is not set on an output compare event. 1 Time-out counter enable. OCF flag is set if an output compare event occurs. TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE.
24:31 CNT	Counter Value These bits reflect the value of a free-running counter used for timeout. **Note:** For proper functionality of this counter, LINIBRR should be greater than or equal to 5. This counter increments once in every bit duration, as per the programmed bit rate. The counter starts as soon as the IP comes out of reset and increments based on the slowest baud rate possible (LINIBRR = FFFFFh), until the user programs LINIBRR and LINFBRR with the desired values.

50.4.2.8 LIN Output Compare Register (LINOCR)

This register contains the value to be compared to the count value in LINTOCR register. This register is writable by software only in Output Compare Mode.

Address: 0x001C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	OC2								OC1							
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 980. LIN Output Compare Register (LINOCR)

Table 1032. LINOCR field descriptions

Field	Description
16:23 OC2	Output compare value 2 When MODE bit = 0 in LINTCSR register: This field is loaded by HW, with response timeout calculated based on RTO value in LINTOCR register and CNT value in LINTCSR register. When MODE bit = 1 in LINTCSR register: This field can be used as output compare register, up to bit rates of 1 Mbit/s.
24:31 OC1	Output compare value 1 When MODE bit = 0 in LINTCSR register: This field is loaded by HW, with header timeout calculated based on HTO value in LINTOCR register and CNT value in LINTCSR register. When MODE bit = 1 in LINTCSR register: This field can be used as output compare register, up to bit rates of 1 Mbit/s. **Note:** OC1 field can be used as output compare register only in LIN slave mode.

50.4.2.9 LIN Time-Out Control Register (LINTOCR)

Address: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	RTO				0	HTO						
W																
Reset	0	0	0	0	1	1	1	0	0	0	0[1]/1	1[1]/0	1	1	0	0

1. This reset value is valid only in Master mode (slave = 0).

Figure 981. LIN Time-Out Control Register (LINTOCR)

Table 1033. LINTOCR field descriptions

Field	Description
20:23 RTO	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0xE = 14, corresponding to $T_{Response\_Maximum} = 1.4 * T_{Response\_Nominal}$
25:31 HTO	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. The reset value is 0x2C = 45, corresponding to $T_{Header\_Maximum}$ in the LIN specification. If Master mode is enabled then these bits will always reflect 0x1C = 28 (1.4 × 10 bits of sync + 1.4 × 10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

50.4.2.10 LIN Fractional Baud Rate Register (LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

Note: When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

This register cannot be used when reduced oversampling is enabled (ROSE bit = 1)

Address: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FBR(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 982. LIN Fractional Baud Rate Register (LINFBRR)

Table 1034. LINFBRR field descriptions

Field	Description
28:31 FBR	Fractional Baud rates 0x0 Fraction (LDIV): 0 0x1 Fraction (LDIV): 1/16 0x2 Fraction (LDIV): 2/16 0x3 Fraction (LDIV): 3/16 0x4 Fraction (LDIV): 4/16 0x5 Fraction (LDIV): 5/16 0x6 Fraction (LDIV): 6/16 0x7 Fraction (LDIV): 7/16 0x8 Fraction (LDIV): 8/16 0x9 Fraction (LDIV): 9/16 0xA Fraction (LDIV): 10/16 0xB Fraction (LDIV): 11/16 0xC Fraction (LDIV): 12/16 0xD Fraction (LDIV): 13/16 0xE Fraction (LDIV): 14/16 0xF Fraction (LDIV): 15/16

50.4.2.11 LIN Integer Baud Rate Register (LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBRR. It can be programmed only in Initialization mode.

Note: When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	IBR(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IBR(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 983. LIN Integer Baud Rate Register (LINIBRR)

Table 1035. LINIBRR field descriptions

Field	Description
12:31 IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. 0x00000 Mantissa (LDIV): LIN clock disabled 0x00001 Mantissa (LDIV): 1 ... 0xFFFFE Mantissa (LDIV): 1048574 0xFFFFF Mantissa (LDIV): 1048575

50.4.2.12 LIN Checksum Field Register (LINCFR)

*Note: There is a delay between 4 to 6 clock cycles of PBRIDGEx_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.*

This register consists of checksum bits.

Address: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CF							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 984. LIN Checksum Field Register (LINCFR)

Table 1036. LINCFR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>24:31
CF</td><td>Checksum bits
When the LINCR1[CCD] bit is reset these bits are read-only and are calculated by hardware. When the LINCR1[CCD] bit is set, these bits can be written by software.
<table>
<tr><th>LINCR1[CFD]</th><th>LINCR1[CCD]</th><th>LINCHKSUM Read/write</th><th>Checksum sent</th></tr>
<tr><td>1</td><td>1</td><td>read/write</td><td>None</td></tr>
<tr><td>1</td><td>0</td><td>read-only</td><td>None</td></tr>
<tr><td>0</td><td>1</td><td>read/write</td><td>Programmed checksum</td></tr>
<tr><td>0</td><td>0</td><td>read-only</td><td>Calculated checksum</td></tr>
</table>
</td></tr>
</table>

50.4.2.13 LIN Control Register 2 (LINCR2)

This register includes control status bits related to buffer operations.

Address: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBDE(1)	IOBE(1)	IOPE(1)	WURQ(2)	DDRQ(2)	DTRQ(2)	ABRQ(2)	HTRQ(2)	0	0	0	0	0	0	0	0
W																
Reset	0	1	0(3)/1	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in Initialization mode.
2. This field can be read/set by software.
3. When slave = 0, reset value is '0' and the this field cannot be programmed.

Figure 985. LIN Control Register 2 (LINCR2)

Table 1037. LINCR2 field descriptions

Field	Description
16 TBDE	Two Bit delimiter bit 0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits This bit can be set in Initialization mode only.
17 IOBE	Idle on Bit Error 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine This bit can be set in Initialization mode only.
18 IOPE	Idle on Identifier Parity Error 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine This bit can be set in Initialization mode only. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request.
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered.
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete.
22 ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also.
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header. Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. **Note:** In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.

50.4.2.14 Buffer Identifier Register (BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

Note: All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Address: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL	DFL	DFL	DFL	DFL	DFL	DIR	CCS	0	0	ID	ID	ID	ID	ID	ID
W	DFL	DFL	DFL	DFL	DFL	DFL	DIR	CCS			ID	ID	ID	ID	ID	ID
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 986. Buffer Identifier Register (BIDR)

Table 1038. BIDR field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes – 1 **Note:** DFL[5:3] is provided for extended frames. Normally LIN mode will use DFL[2:0]. Identifier filters are now compatible with DFL[5:0] also.
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copies them in the BDRs 1 LINFlexD transmits the data from the BDRs
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = ‘1’).

50.4.2.15 Buffer Data Register Least Significant (BDRL)

This register is a part of an 8-byte data buffer.

Address: 0x0038 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA3								DATA2							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 987. Buffer Data Register Least Significant (BDRL)

Table 1039. BDRL field descriptions

Field	Description
0:7 DATA3	Data Byte 3 Data byte 3 of the data field
8:15 DATA2	Data Byte 2 Data byte 2 of the data field
16:23 DATA1	Data Byte 1 Data byte 1of the data field
24:31 DATA0	Data Byte 0 Data byte 0 of the data field

50.4.2.16 Buffer Data Register Most Significant (BDRM)

This register is a part of an 8-byte data buffer.

Address: 0x003C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA7								DATA6							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA5								DATA4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 988. Buffer Data Register Most Significant (BDRM)

Table 1040. BDRM field descriptions

Field	Description
0:7 DATA7	Data Byte 7 Data byte 7 of the data field
8:15 DATA6	Data Byte 6 Data byte 6 of the data field
16:23 DATA5	Data Byte 5 Data byte 5 of the data field
24:31 DATA4	Data Byte 4 Data byte 4 of the data field

50.4.2.17 Identifier Filter Enable Register (IFER)

This register enables/disables a particular filter.

Address: 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FACT(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 989. Identifier Filter Enable Register (IFER)

Table 1041. IFER field descriptions

Field	Description
16:31 FACT	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. 1. In Identifier mask mode, FACT (2*n*+1) have no effect on the corresponding filters as they act as mask for the Identifier 2*n*. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. **Note:** Refer to device configuration chapter to see the number of filters used in the device.

50.4.2.18 Identifier Filter Match Index (IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Address: 0x0044 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	IFMI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 990. Identifier Filter Match Index (IFMI)

Table 1042. IFMI field descriptions

Field	Description
27:31 IFMI	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (refer to *Section 50.3.2.4* for more details).

50.4.2.19 Identifier Filter Mode Register (IFMR)

This register configures the modes of filters.

Address: 0x0048 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	IFM(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 991. Identifier Filter Mode Register (IFMR)

Table 1043. IFMR field descriptions

Field	Description
24:31 IFM	Filter mode 0 Filters 2*n* and 2*n*+1 are in identifier list mode 1 Filters 2*n* and 2*n*+1 are in mask mode, where filter 2*n*+1 is the mask for filter 2*n*

50.4.2.20 Identifier Filter Control Register (IFCR2*n*)

This register is read-only in normal mode and can be programmed only in Initialization mode. This register acts as a filter in both Identifier list and Identifier mask modes.

Address: 0x4C+ *n**0x8 (*n* = 0 to (no_of_filters[2]/2–1)) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL[1]						DIR[1]	CCS[1]	0	0	ID[1]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. Refer to the Device Configuration chapter to see the number of filters used in this device.

Figure 992. Identifier Filter Control Register (IFCR2n)

Table 1044. IFCR2*n* field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes - 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity.

50.4.2.21 Identifier Filter Control Register (IFCR2*n*+1)

This register is read-only in normal mode and can be programmed only in Initialization mode. This register acts as a filter in Identifier list mode but acts as a mask for IFCR2*n* in Identifier mask mode.

Address: 0x50+ *n**0x8 (*n* = 0 to (no_of_filters(2)/2–1)) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL(1)						DIR(1)	CCS(1)	0	0	ID(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. Refer to the Device Configuration chapter to see the number of filters used in this device.

Figure 993. Identifier Filter Control Register (IFCR2n+1)

Table 1045. IFCR2*n*+1 field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes – 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity.

50.4.2.22 Global Control Register (GCR)

This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

Address: 0x8C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDFBM(1)	RDFBM(1)	TDLIS(1)	RDLIS(1)	STOP(1)	SR(2)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. This field is read 0 by software.

Figure 994. Global Control Register (GCR)

Table 1046. GCR field descriptions

Field	Description
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. 0 The first bit of received data is LSB – in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB – in other words, the first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. 0 Received data is not inverted 1 Received data is inverted

Table 1046. GCR field descriptions (continued)

Field	Description
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). 0 1 stop bit 1 2 stop bits
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). This bit is always read as 0. The following register fields are reset by SR: LINSR: All fields except RXbusy and AUTOSYNC_COMP LINESR: All fields LINTCSR: Only CNT[0:7] UARTSR: All fields except RFE and TFFt UARTCR: Only TFC and RFC UARTCTO: All fields

50.4.2.23 UART Preset Timeout Register (UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be received or to monitor the idle state of the reception line. This register can be written by software any time.

Address: 0x8C + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PTO											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 995. UART Preset Timeout Register (UARTPTO)

Table 1047. UARTPTO field descriptions

Field	Description
20:31 PTO	Preset Timeout PTO defines the preset value of timeout counter. A value of 0, 1 or 2 is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer to the UARTCTO register.

50.4.2.24 UART Current Timeout Register (UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (refer to *Section 50.4.2.23*) to monitor the number of bits received by UART or to monitor the idle state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGEx_CLK (when ROSE = 0)
- Is clocked with LIN_CLK / (OSR * IBRR) synchronized to PBRIDGEx_CLK (when ROSE = 1)
- Is automatically enabled when UARTCR[RXEN] = 1

*Note: Due to this synchronisation, UARTCTO reflects the internal counter's value (which is clocked with LIN_CLK/16 * LDIV when ROSE = 0 or LIN_CLK/OSR * IBRR when ROSE = 1) which was present on it, before 4 to 6 clock cycles of PBRIDGEx_CLK.*

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO-1
- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

Address: 0x8C + 0x08 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CTO											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 996. UART Current Timeout Register (UARTCTO)

Table 1048. UARTCTO field descriptions

Field	Description
20:31 CTO	Current Timeout CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

50.4.2.25 DMA Tx Enable Register (DMATXE)

This register enables the DMA TX interface. This register can be written and read by software any time.

Address: 0x8C + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	DTE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 997. DMA Tx Enable Register (DMATXE)

Table 1049. DMATXE field descriptions

Field	Description
(32-2^TX_CH_NUM):31 DTE[(2^TX_CH_NUM-1):0]	DMA Tx channel Y enable 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled **Note:** The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. Refer to the device configuration chapter for the value of TX_CH_NUM used in this device.

50.4.2.26 DMA Rx Enable Register (DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

Address: 0x8C + 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	DRE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 998. DMA Rx Enable Register (DMARXE)

Table 1050. DMARXE field descriptions

Field	Description
(32–2^RX_CH_NUM):31 DRE[(2^RX_CH_NUM-1):0]	DMA Rx channel Y enable 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled **Note:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. Refer to the device configuration chapter for the value of RX_CH_NUM used in this device.

50.5 Programming considerations

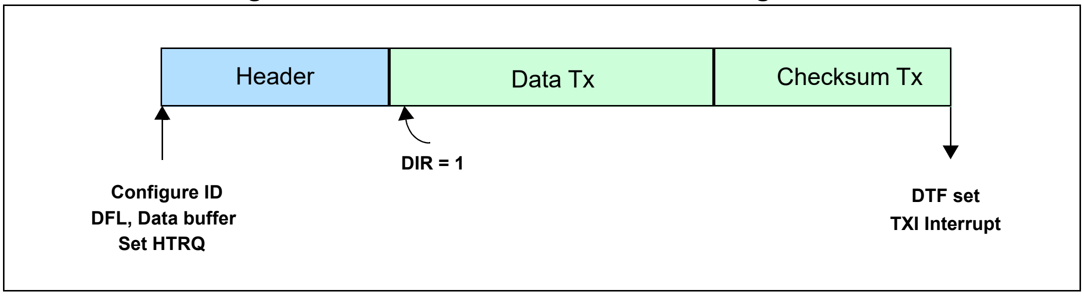
The next subsections describe the various configurations in which the LINFlexD module can be used.

50.5.1 Master node

LINFlexD acts as Master when the MME bit is set.

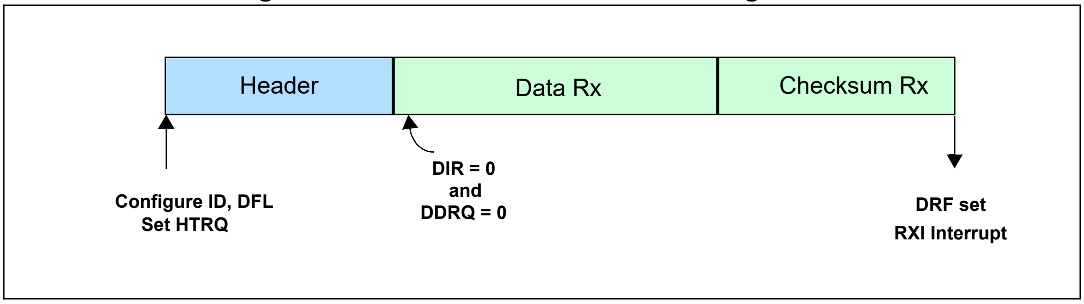
50.5.1.1 Transmitter

Figure 999. Master node: transmitter configuration



50.5.1.2 Receiver

Figure 1000. Master node: receiver configuration



50.5.1.3 Transmitter, Bit Error

Figure 1001. Master node: transmitter, bit error configuration

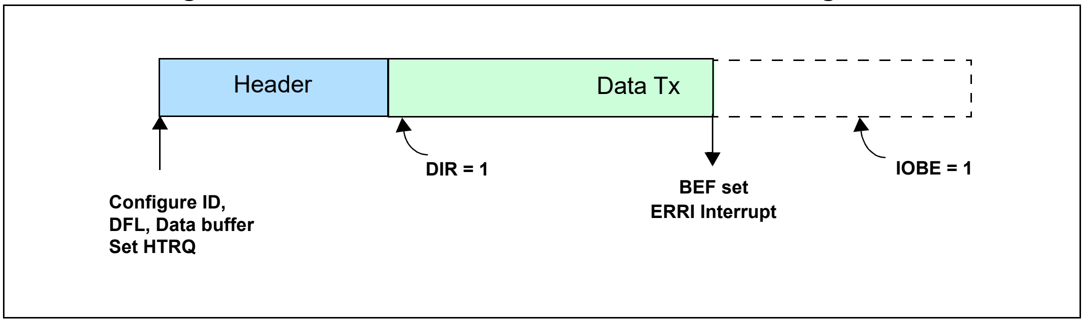
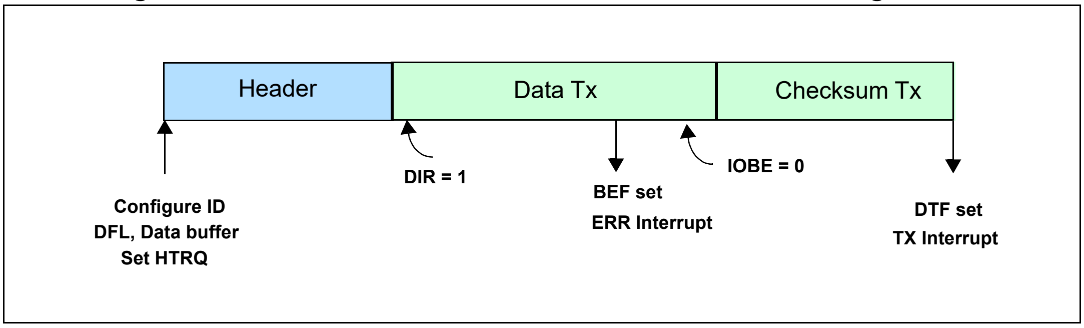
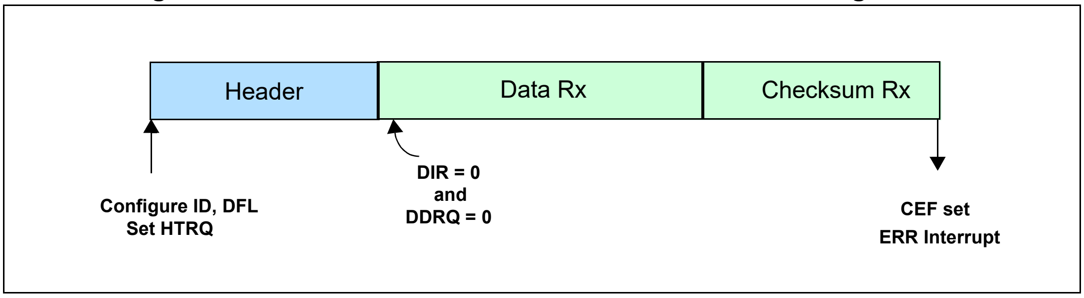


Figure 1002. Master node: transmitter, checksum error configuration



50.5.1.4 Receiver, Checksum Error

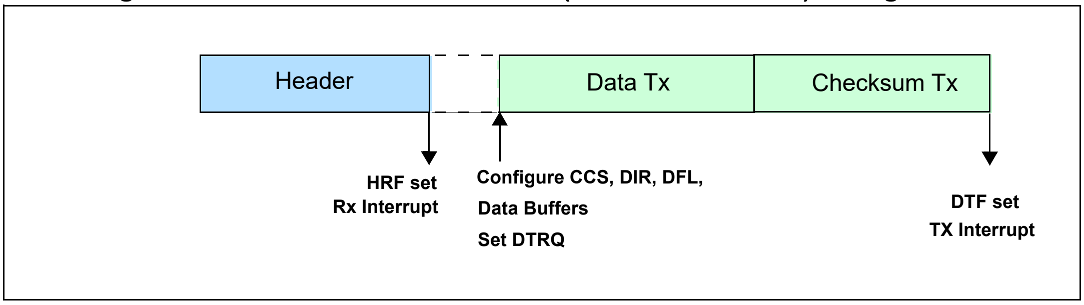
Figure 1003. Master node: receiver, checksum error configuration



50.5.2 Slave node

50.5.2.1 Transmitter (no identifier filters)

Figure 1004. Slave node: transmitter (no identifier filters) configuration



50.5.2.2 Receiver (no identifier filters)

Figure 1005. Slave node: receiver (no identifier filters) configuration

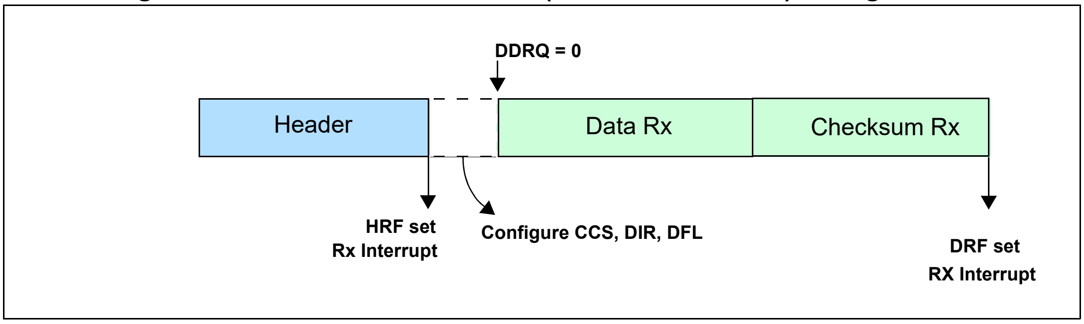
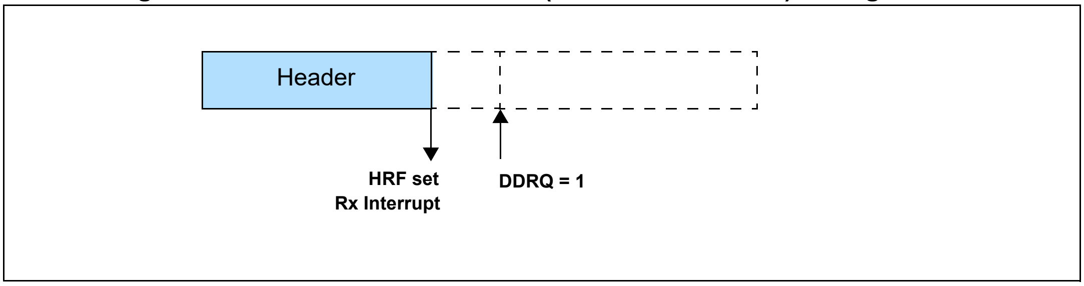
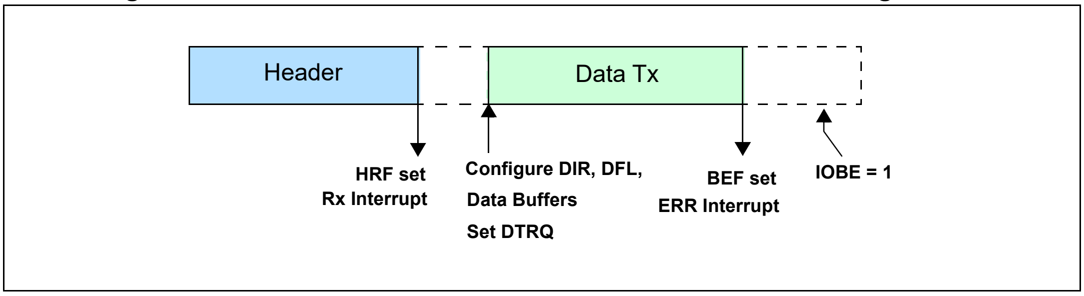


Figure 1006. Slave node: receiver (no identifier filters) configuration



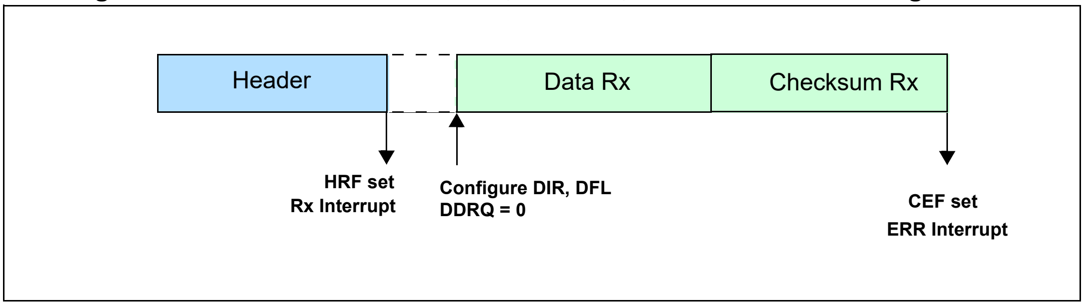
50.5.2.3 No filters, transmitter, bit error

Figure 1007. Slave node: No filters, transmitter, bit error configuration



50.5.2.4 No filters, receiver, checksum error

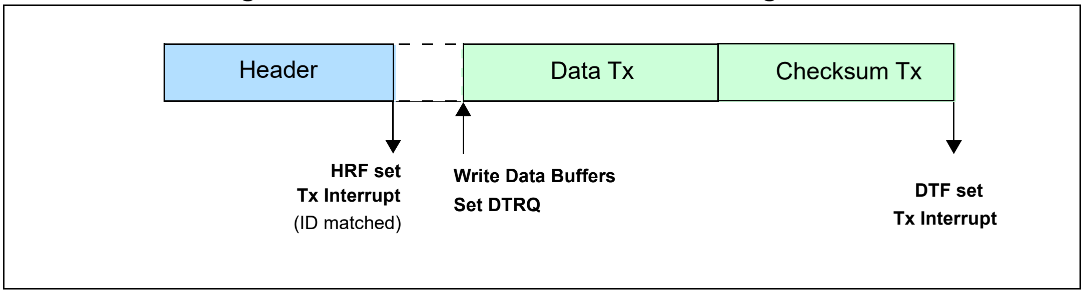
Figure 1008. Slave node: No filters, receiver, checksum error configuration



50.5.2.5 At least one TX filter, BF is reset, ID matches filter

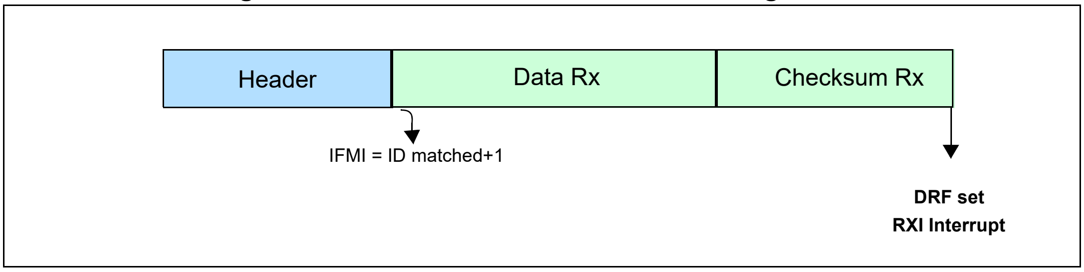
This configuration can be used in case slave never receives data, for example, sensor.

Figure 1009. Slave node: one TX filter configuration



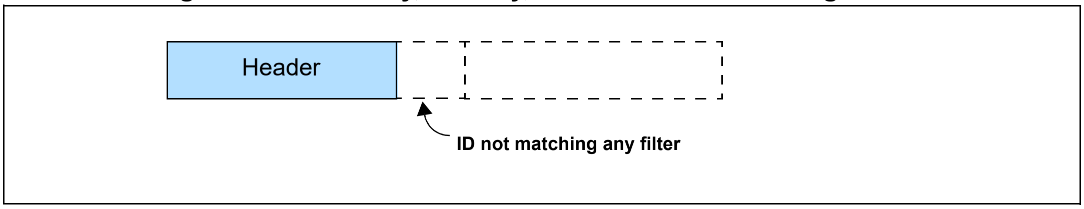
50.5.2.6 At least one RX filter, BF reset, ID matches filter

Figure 1010. Slave node: one RX filter configuration



50.5.2.7 RX only, TX only, RX and TX filters, ID not matching filter, BF reset

Figure 1011. RX only, TX only, RX and TX filters configuration



50.5.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

Figure 1012. TX filter, BF is set configuration: ID has matched

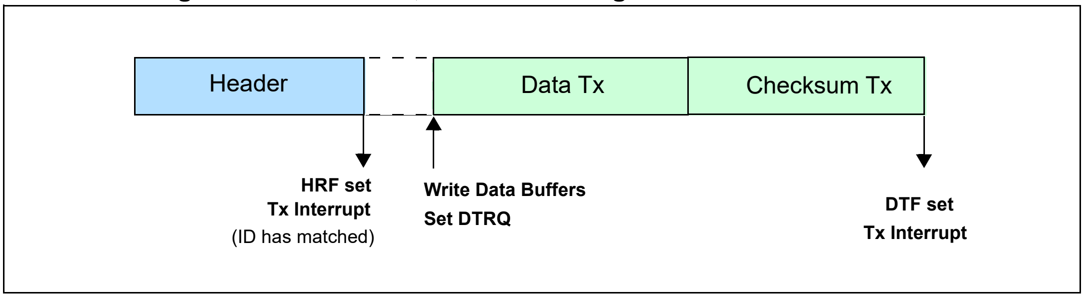
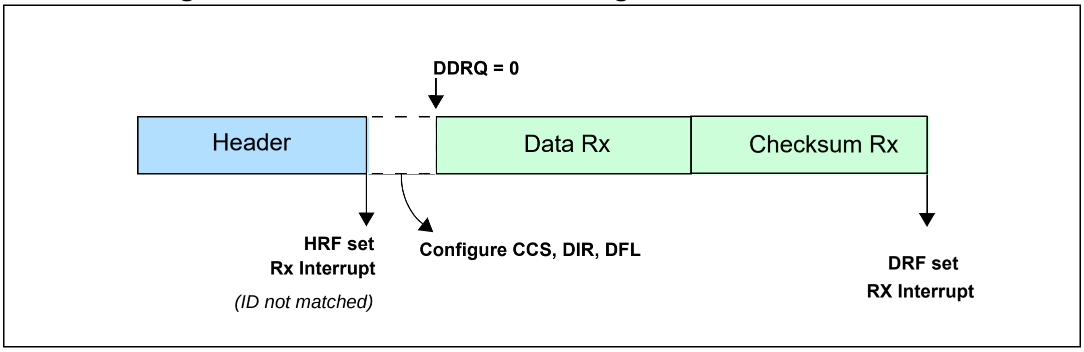


Figure 1013. TX filter, BF is set configuration: ID not matched



50.5.2.9 RX filter, BF is set

Figure 1014. RX filter, BF is set configuration: ID matched

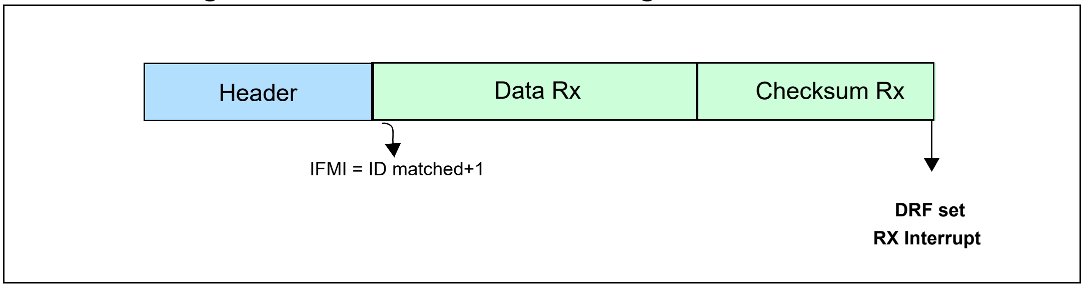


Figure 1015. RX filter, BF is set configuration: ID not matched (ID is Rx)

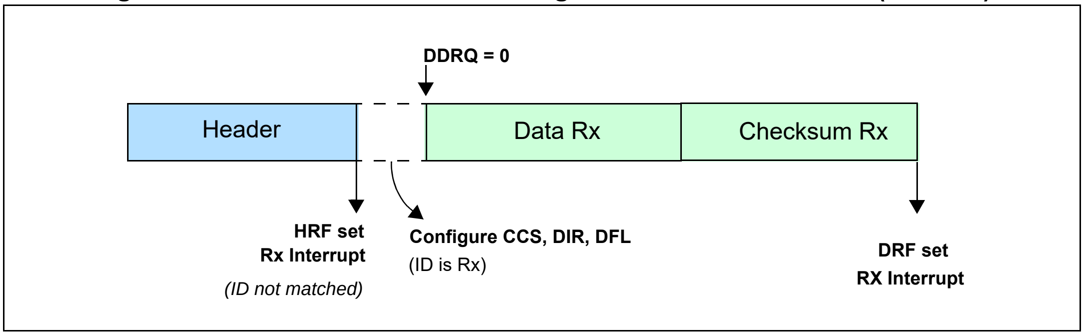
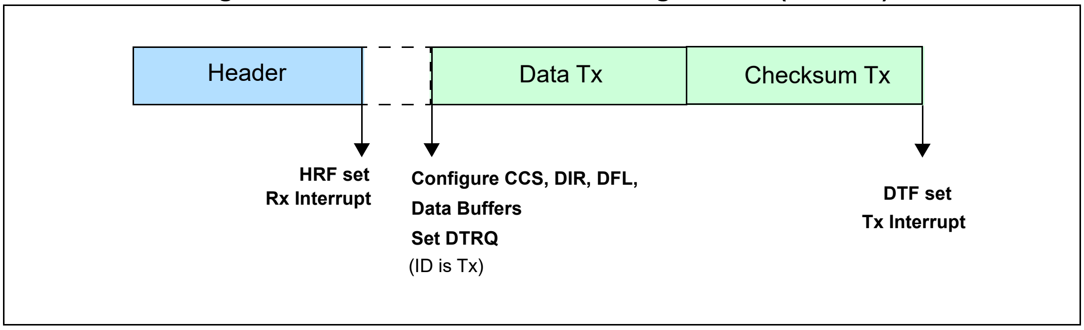


Figure 1016. RX filter, BF is set configuration: (ID is Tx)



50.5.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

Figure 1017. TX filter, RX filter, BF set configuration: (ID matched)

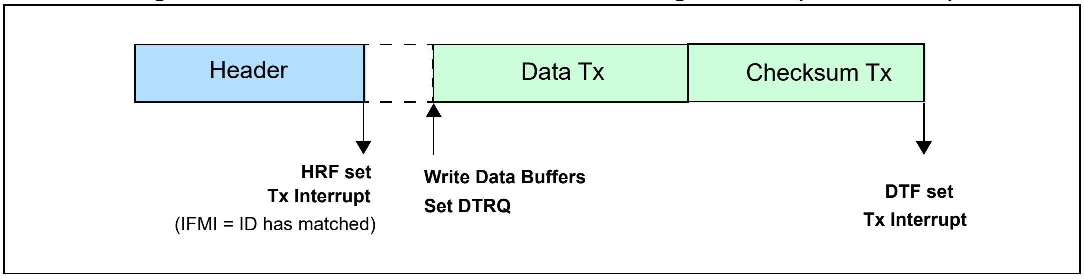


Figure 1018. TX filter, RX filter, BF set configuration: (ID matched + 1)

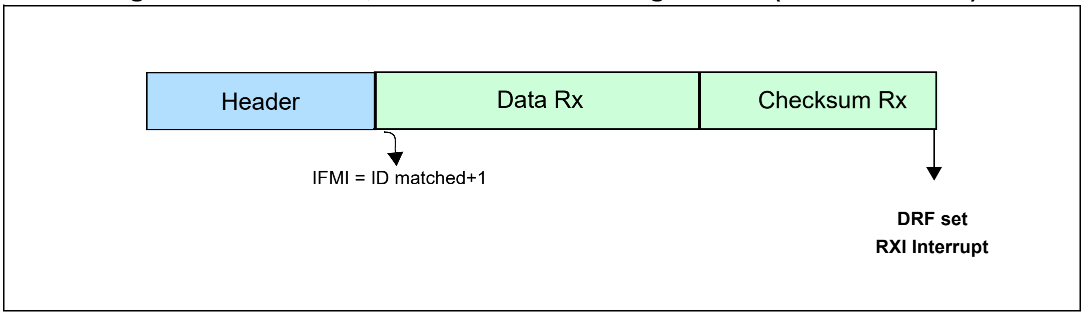
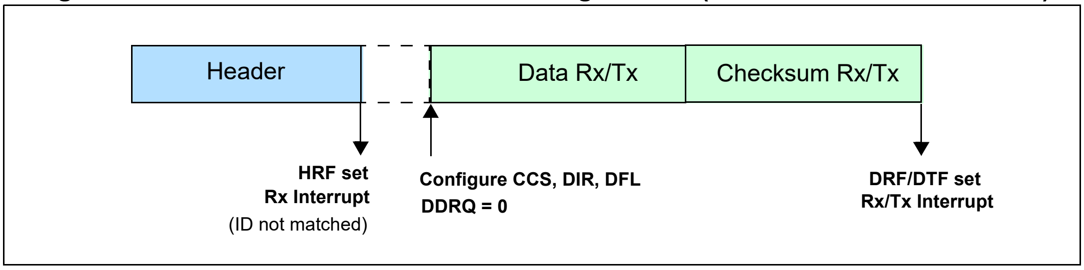


Figure 1019. TX filter, RX filter, BF set configuration: (ID not matched, RX/TX Int.)



50.5.3 Extended frames

Figure 1020. Extended frames (TX Interrupt)

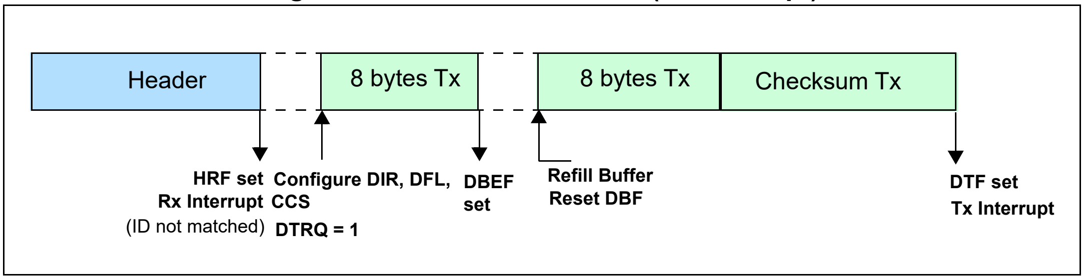
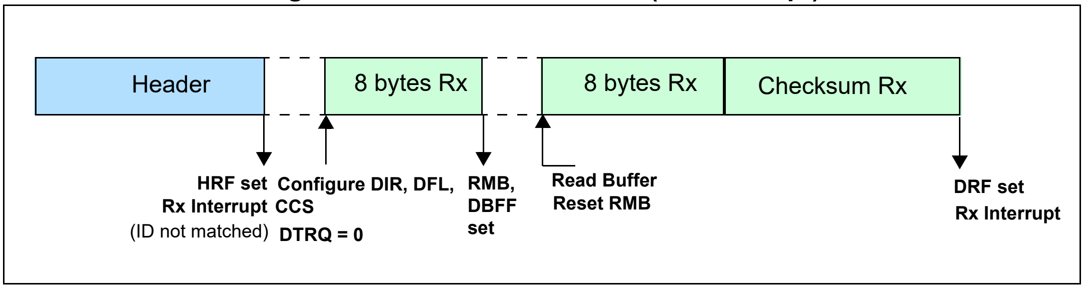


Figure 1021. Extended frames (RX Interrupt)



50.5.4 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

Figure 1022. Response timeout

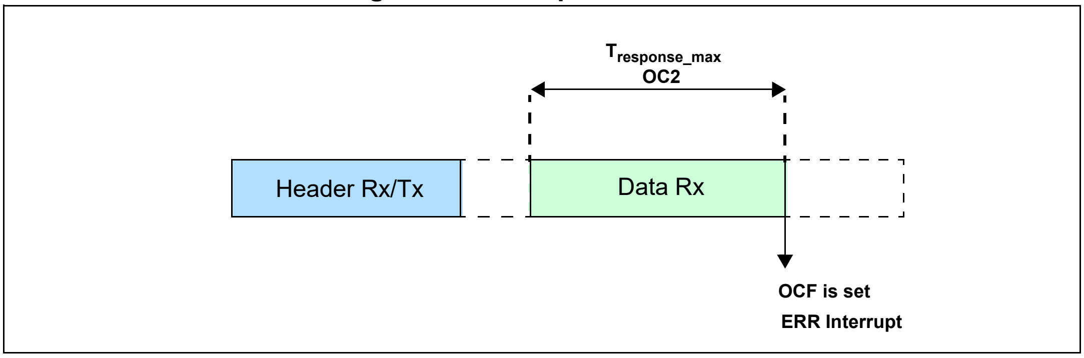
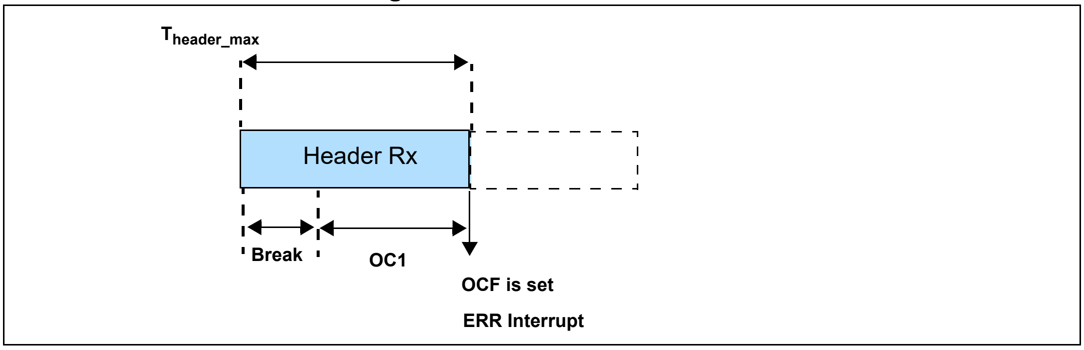
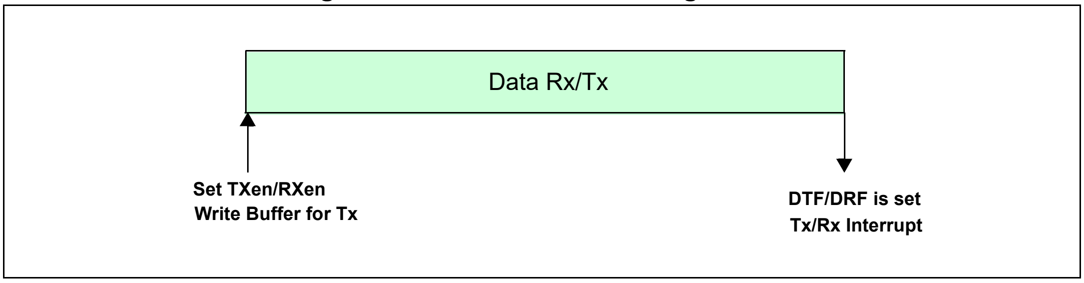


Figure 1023. Header timeout



50.5.5 UART mode

Figure 1024. UART mode configuration



50.5.6 Interrupts

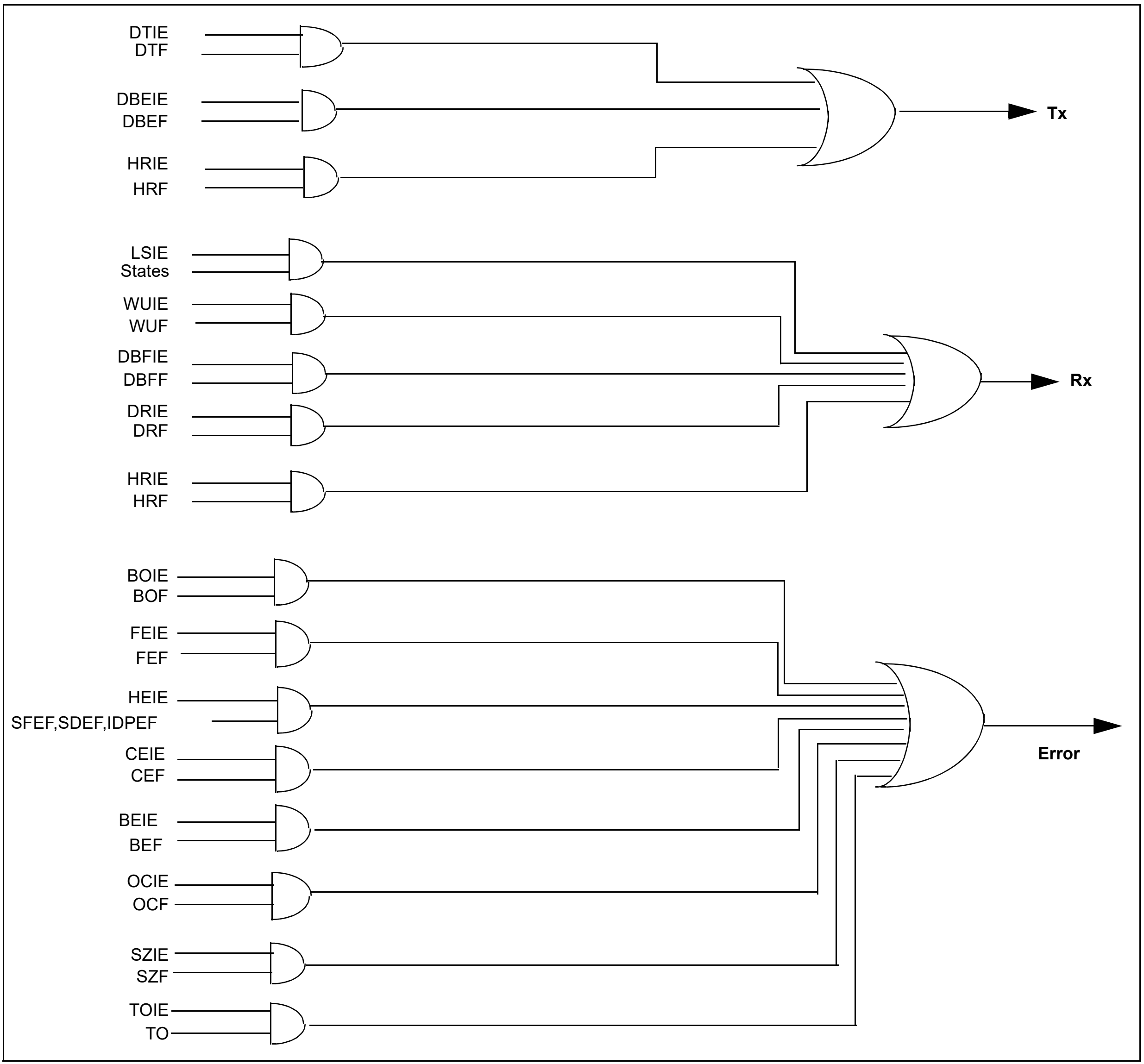
Table 1051. Interrupts

Interrupt event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error
Header error	SFEF or SDEF or IDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer overrun error	BOF	BOIE	Error
UART timeout error	TO	TOIE	Error
LIN state	Sync Del, Sync Field, Identifier Field, or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data buffer full	DBFF	DBFIE	Rx

Table 1051. Interrupts (continued)

Interrupt event	Event flag	Enable control bit	Interrupt vector
Data buffer empty	DBEF	DBEIE	Tx
Data reception complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

Figure 1025. Interrupt diagram



51 Reset Generation Module (MC_RGM)

51.1 Introduction

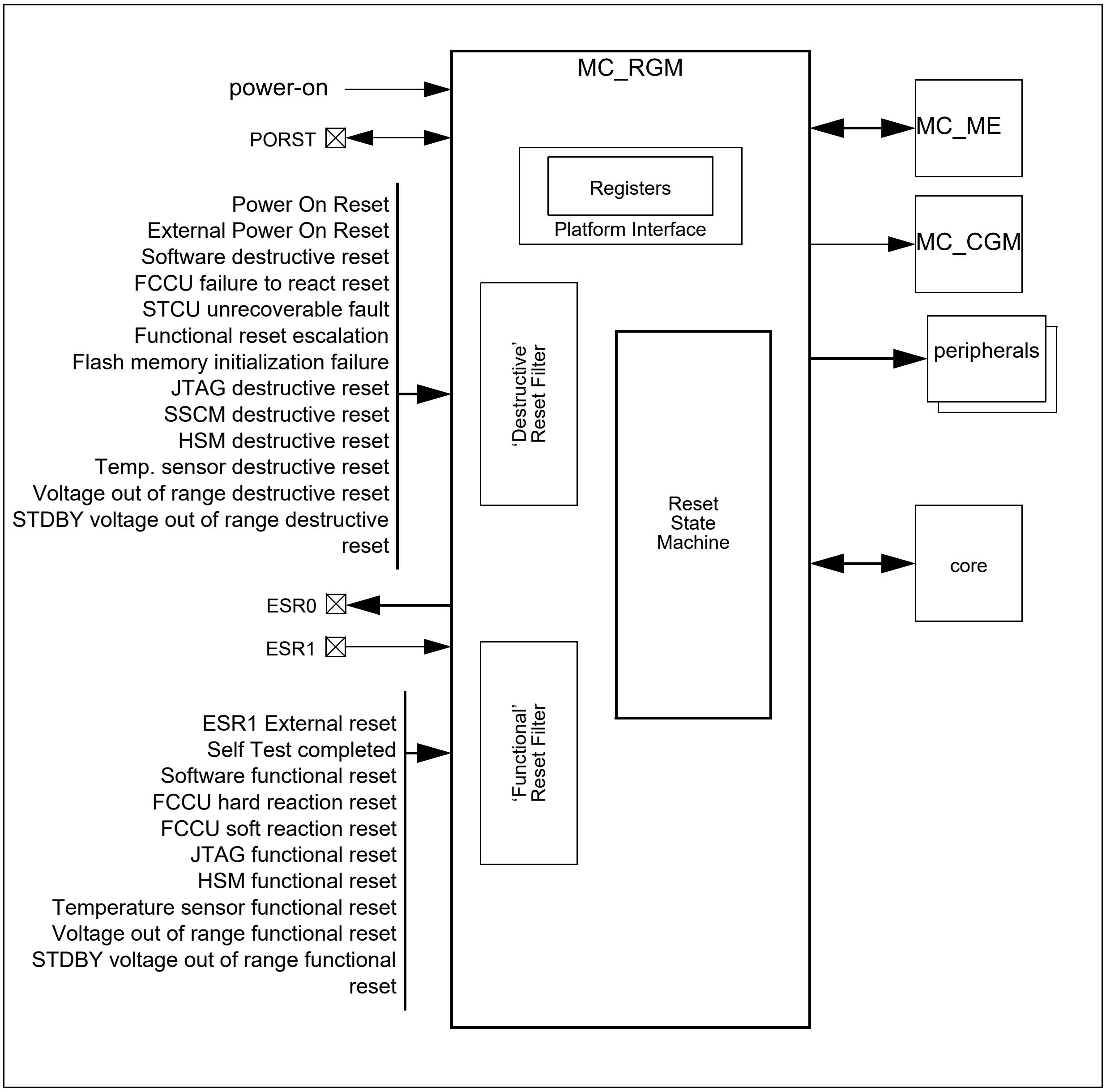
This chapter describes the MC_RGM module.

51.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

Figure 1026 shows the MC_RGM block diagram.

Figure 1026. MC_RGM block diagram



51.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'destructive' resets management
- 'functional' resets management
- signalling of reset events after each reset sequence (reset status flags)
- conversion of reset events to SAFE mode or interrupt request events
- short reset sequence configuration
- bidirectional reset behavior configuration

51.1.3 Reset sources

The different reset sources are organized into two families: ‘destructive’ and ‘functional’.

- A ‘destructive’ reset source is associated with an event related to a critical (usually hardware) error or dysfunction. When a ‘destructive’ reset event occurs, the full reset sequence is applied to the chip starting from PHASE0. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be unknown .
  ‘Destructive’ resets are:
  - Power On Reset
  - External Power On Reset
  - Software destructive reset
  - FCCU failure to react reset
  - STCU unrecoverable fault
  - Functional reset escalation
  - Flash memory initialization failure
  - JTAG destructive reset
  - SSCM destructive reset
  - HSM destructive reset
  - Temperature sensor destructive reset
  - Voltage out of range destructive reset
  - STDBY voltage out of range destructive reset
- A ‘functional’ reset source is associated with an event related to a less-critical (usually non-hardware) error or dysfunction. When a ‘functional’ reset event occurs, a partial reset sequence is applied to the chip starting from PHASE1. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules (for example, debug modules, flash modules) as well as the system memory content is preserved, except on an external reset in the case where the chip is configured to execute a self-test on power-on, ‘destructive’, and external resets.
  ‘Functional’ resets are:
  - ESR1 External reset
  - Self Test completed
  - Software functional reset
  - FCCU hard reaction reset
  - FCCU soft reaction reset
  - JTAG functional reset
  - HSM functional reset
  - Temperature sensor functional reset
  - Voltage out of range functional reset
  - STDBY voltage out of range functional reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (that is PHASE*n* states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the IDLE phase. During this entire process, the MC_ME state machine is held in RESET mode. Only at the end of the reset sequence, when the IDLE phase is reached, does the MC_ME enter the DRUN mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt issued to the core (refer to *Section 51.3.1.2: 'Destructive' Event Reset Disable Register (RGM_DERD)* and *Section 51.3.1.3: 'Destructive' Event Alternate Request Register (RGM_DEAR)* for 'destructive' resets and *Section 51.3.1.6: 'Functional' Event Reset Disable Register (RGM_FERD)* and *Section 51.3.1.7: 'Functional' Event Alternate Request Register (RGM_FEAR)* for 'functional' resets).

51.2 External signal description

The MC_RGM interfaces to the output reset pin ESR0.

51.3 Memory map and register definition

Table 1052. MC_RGM memory map

Address offset	Name	Description	Location
0x000	RGM_DES	'Destructive' Event Status	*Section 51.3.1.1*
0x004–0x00F	Reserved		
0x010	RGM_DERD	'Destructive' Event Reset Disable	*Section 51.3.1.2*
0x014–0x01F	Reserved		
0x020	RGM_DEAR	'Destructive' Event Alternate Request	*Section 51.3.1.3*
0x024–0x02F	Reserved		
0x030	RGM_DBRE	'Destructive' Reset Enable	*Section 51.3.1.4*
0x034–0x2FF	Reserved		
0x300	RGM_FES	'Functional' Event Status	*Section 51.3.1.5*
0x304–0x30F	Reserved		
0x310	RGM_FERD	'Functional' Event Reset Disable	*Section 51.3.1.6*
0x314–0x31F	Reserved		
0x320	RGM_FEAR	'Functional' Event Alternate Request	*Section 51.3.1.7*
0x324–0x32F	Reserved		
0x330	RGM_FBRE	'Functional' Reset Enable	*Section 51.3.1.8*
0x334–0x33F	Reserved		
0x340	RGM_FESS	'Functional' Event Short Sequence	*Section 51.3.1.9*
0x344–0x603	Reserved		
0x604	RGM_FRET	'Functional' Reset Threshold	*Section 51.3.1.10*
0x605–0x607	Reserved		

Table 1052. MC_RGM memory map (continued)

Address offset	Name	Description	Location
0x608	RGM_DRET	'Destructive' Reset Escalation Threshold Register	Section 51.3.1.11
0x609–0x60F	Reserved		
0x610	RGM_PRST0	Peripheral Reset 0	Section 51.3.1.12
0x614	RGM_PRST1	Peripheral Reset 1	Section 51.3.1.13
0x618	RGM_PRST2	Peripheral Reset 2	Section 51.3.1.14
0x61C	RGM_PRST3	Peripheral Reset 3	Section 51.3.1.15
0x620	RGM_PRST4	Peripheral Reset 4	Section 51.3.1.16
0x624	RGM_PRST5	Peripheral Reset 5	Section 51.3.1.17
0x628	RGM_PRST6	Peripheral Reset 6	Section 51.3.1.18
0x62C	RGM_PRST7	Peripheral Reset 7	Section 51.3.1.19
0x630	RGM_PSTAT0	Peripheral Reset Status 0	Section 51.3.1.20
0x634	RGM_PSTAT1	Peripheral Reset Status 1	Section 51.3.1.21
0x638	RGM_PSTAT2	Peripheral Reset Status 2	Section 51.3.1.22
0x63C	RGM_PSTAT3	Peripheral Reset Status 3	Section 51.3.1.23
0x640	RGM_PSTAT4	Peripheral Reset Status 4	Section 51.3.1.24
0x644	RGM_PSTAT5	Peripheral Reset Status 5	Section 51.3.1.25
0x648	RGM_PSTAT6	Peripheral Reset Status 6	Section 51.3.1.26
0x64C	RGM_PSTAT7	Peripheral Reset Status 7	Section 51.3.1.27

Note: Any access to unused registers as well as write accesses to read-only registers will:

- *not change register content*
- *cause a transfer error.*

51.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the RGM_DES[8:15] register bits may be accessed as a word at address offset 0x000, as a half-word at address offset 0x002, or as a byte at address offset 0x003.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

51.3.1.1 'Destructive' Event Status Register (RGM_DES)

This register contains the status of the 'destructive' reset sources, including those configured not to generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on writing '1'.

This register is reset only on power-on.

Address: 0x000 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	F_VOR_STDBY	F_VOR_DEST	F_TSR_DEST	0	0	0	0	0	0	0
W							w1c	w1c	w1c							
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_HSM_DEST	F_SSCM_DEST	0	0	0	F_JTAG_DEST	F_FIF	F_EDR	0	0	F_SUF	F_FFRR	F_SOFT_DEST	0	F_PORST	F_POR
W	w1c	w1c				w1c	w1c	w1c			w1c	w1c	w1c		w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1027. 'Destructive' Event Status Register (RGM_DES)

Table 1053. RGM_DES field descriptions

Field	Description
6 F_VOR_STDBY	Flag for voltage out of range from LVDs or HVDs, or both, during standby mode 'destructive' reset 0 No voltage out of range 'destructive' reset event has occurred since the last clear 1 A voltage out of range 'destructive' reset event has occurred **Note:** for LVD100_SB it is not allowed to enable destructive reset, only functional reset is supported.
7 F_VOR_DEST	Flag for voltage out of range 'destructive' reset 0 No voltage out of range 'destructive' reset event has occurred since the last clear 1 A voltage out of range 'destructive' reset event has occurred
8 F_TSR_DEST	Flag for temperature sensor 'destructive' reset 0 No temperature sensor 'destructive' reset event has occurred since the last clear 1 A temperature sensor 'destructive' reset event has occurred
16 F_HSM_DEST	Flag for HSM 'destructive' reset request 0 No HSM 'destructive' reset request event has occurred since the last clear 1 A HSM 'destructive' reset request event has occurred
17 F_SSCM_DEST	Flag for SSCM 'destructive' reset request 0 No SSCM 'destructive' reset request event has occurred since the last clear 1 A SSCM 'destructive' reset request event has occurred

Table 1053. RGM_DES field descriptions (continued)

Field	Description
21 F_JTAG_DEST	Flag for JTAG 'destructive' reset request 0 No JTAG 'destructive' reset request event has occurred since the last clear 1 A JTAG 'destructive' reset request event has occurred
22 F_FIF	Flag for flash memory initialization failure 0 No flash memory initialization failure event has occurred since the last clear 1 A flash memory initialization failure event has occurred
23 F_EDR	Flag for ‘functional’ reset escalation 0 No ‘functional’ reset escalation event has occurred since the last clear 1 A ‘functional’ reset escalation event has occurred
26 F_SUF	Flag for STCU2 unrecoverable fault 0 No STCU2 unrecoverable fault event has occurred since the last clear 1 A STCU2 unrecoverable fault event has occurred
27 F_FFRR	Flag for FCCU failure to react reset 0 No FCCU failure to react reset event has occurred since the last clear 1 A FCCU failure to react reset event has occurred
28 F_SOFT_DEST	Flag for software ‘destructive’ reset 0 No software ‘destructive’ reset event has occurred since the last clear 1 A software ‘destructive’ reset event has occurred
30 F_PORST	Flag for external power-on reset 0 No external power-on reset has occurred since the last clear 1 A external power-on reset event has occurred **Note:** F_PORST will not be set if an external PORST is asserted during standby mode. Rather MC_RGM.FES.F_JTAG_FUNC will be set.
31 F_POR	Flag for Power-On reset 0 No power-on event has occurred since the last clear 1 A power-on event has occurred

51.3.1.2 ‘Destructive’ Event Reset Disable Register (RGM_DERD)

This register provides dedicated bits to disable particular ‘destructive’ reset sources. When a ‘destructive’ reset source is disabled, the associated ‘destructive’ event will trigger either a SAFE mode request or an interrupt request (refer to *Section 51.3.1.3: ‘Destructive’ Event Alternate Request Register (RGM_DEAR)*). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after a ‘destructive’ or power-on reset.

This register is reset only on power-on.

Caution: It is important to clear the RGM_DES register before setting any of the bits in the RGM_DERD register to ‘1’. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0x010 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	D_VOR_STDBY	D_VOR_DEST	D_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	D_HSM_DEST	D_SSCM_DEST	0	0	0	D_JTAG_DEST	D_FIF	D_EDR	0	0	D_SUF	D_FFRR	D_SOFT_DEST	0	D_PORST	D_POR
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1028. ‘Destructive’ Event Reset Disable Register (RGM_DERD)

Table 1054. RGM_DERD field descriptions

Field	Description
6 D_VOR_STDBY	Disable voltage out of range from LVDs or HVDs, or both, during standby mode ‘destructive’ reset 0 A voltage out of range ‘destructive’ reset event triggers a reset sequence
7 D_VOR_DEST	Disable voltage out of range ‘destructive’ reset 0 A voltage out of range ‘destructive’ reset event triggers a reset sequence
8 D_TSR_DEST	Disable temperature sensor 'destructive' reset 0 A temperature sensor 'destructive' reset event triggers a reset sequence
16 D_HSM_DEST	Disable HSM 'destructive' reset request 0 A HSM 'destructive' reset request event triggers a reset sequence
17 D_SSCM_DEST	Disable SSCM 'destructive' reset request 0 A SSCM 'destructive' reset request event triggers a reset sequence
21 D_JTAG_DEST	Disable JTAG 'destructive' reset request 0 A JTAG 'destructive' reset request event triggers a reset sequence
22 D_FIF	Disable flash memory initialization failure 0 A flash memory initialization failure event triggers a reset sequence
23 D_EDR	Disable ‘functional’ reset escalation 0 A ‘functional’ reset escalation event triggers a reset sequence
26 D_SUF	Disable STCU2 unrecoverable fault 0 A STCU2 unrecoverable fault event triggers a reset sequence
27 D_FFRR	Disable FCCU failure to react reset 0 A FCCU failure to react reset event triggers a reset sequence

Table 1054. RGM_DERD field descriptions (continued)

Field	Description
28 D_SOFT_DEST	Disable software 'destructive' reset 0 A software 'destructive' reset triggers a reset sequence
30 D_PORST	Disable external power-on reset 0 An external power-on reset event triggers a reset sequence 1 An external power-on reset event generates either a SAFE mode or an interrupt request depending on the values in RGM_DEAR.AR_PORST
31 D_POR	Disable Power-On reset 0 A power-on event triggers a reset sequence

51.3.1.3 'Destructive' Event Alternate Request Register (RGM_DEAR)

This register defines an alternate request to be generated when a reset on a 'destructive' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0x020 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AR_PORST	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1029. 'Destructive' Event Alternate Request Register (RGM_DEAR)

Table 1055. RGM_DEAR field descriptions

Field	Description
30 AR_PORST	Alternate Request for external power-on reset 0 Generate a SAFE mode request on an external power-on reset event if the reset is disabled 1 Generate an interrupt request on an external power-on reset event if the reset is disabled

51.3.1.4 'Destructive' Reset Enable Register (RGM_DBRE)

This register enables the generation of an external reset on 'destructive' reset. It can be accessed in read only in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0x030 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	BE_VOR_STDBY	BE_VOR_DEST	BE_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE_HSM_DEST	BE_SSCM_DEST	0	0	0	BE_JTAG_DEST	BE_FIF	BE_EDR	0	0	BE_SUF	BE_FFRR	BE_SOFT_DEST	0	BE_PORST	BE_POR
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1030. ‘Destructive’ Reset Enable Register (RGM_DBRE)

Table 1056. RGM_DBRE field descriptions

Field	Description
6 BE_VOR_STDBY	Reset Enable for voltage out of range from LVDs or HVDs, or both, during standby mode ‘destructive’ reset 0 ESR0 is asserted on a voltage out of range ‘destructive’ reset event
7 BE_VOR_DEST	Reset Enable for voltage out of range ‘destructive’ reset 0 ESR0 is asserted on a voltage out of range ‘destructive’ reset event
8 BE_TSR_DEST	Reset Enable for temperature sensor 'destructive' reset 0 ESR0 is asserted on a temperature sensor 'destructive' reset event if the reset is enabled
16 BE_HSM_DEST	Reset Enable for HSM 'destructive' reset request 0 ESR0 is asserted on a HSM 'destructive' reset request event
17 BE_SSCM_DEST	Reset Enable for SSCM 'destructive' reset request 0 ESR0 is asserted on a SSCM 'destructive' reset request event
21 BE_JTAG_DEST	Reset Enable for JTAG 'destructive' reset request 0 ESR0 is asserted on a JTAG 'destructive' reset request event 1 ESR0 is not asserted on a JTAG 'destructive' reset request event
22 BE_FIF	Reset Enable for flash memory initialization failure 0 ESR0 is asserted on a flash memory initialization failure event
23 BE_EDR	Reset Enable for ‘functional’ reset escalation 0 ESR0 is asserted on a ‘functional’ reset escalation event

Table 1056. RGM_DBRE field descriptions (continued)

Field	Description
26 BE_SUF	Reset Enable for STCU2 unrecoverable fault 0 ESR0 is asserted on a STCU2 unrecoverable fault event
27 BE_FFRR	Reset Enable for FCCU failure to react reset 0 ESR0 is asserted on a FCCU failure to react reset event
28 BE_SOFT_DEST	Reset Enable for software ‘destructive’ reset 0 ESR0 is asserted on a software ‘destructive’ reset event
30 BE_PORST	Reset Enable for external power-on reset 0 ESR0 is asserted on an external power-on reset event if the reset is enabled
31 BE_POR	Reset Enable for power-on reset 0 ESR0 is asserted on a power-on reset event

Note: For more details about ESR0, refer to Section 51.4.3.2: ESR0 Output External reset.

51.3.1.5 ‘Functional’ Event Status Register (RGM_FES)

This register contains the status of the ‘functional’ reset sources, including those configured not to generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on writing ‘1’ if the triggering event has already been cleared at the source.

This register is reset only on power-on.

Note: If a ‘functional’ reset source is configured to generate a SAFE mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles before it clears the associated RGM_FES status bit in order to avoid multiple SAFE mode requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the RGM_FES has been properly cleared, and if not, clear it again.

Address: 0x300 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	F_VOR_FUNC_STDBY	F_VOR_FUNC	F_TSR_FUNC	0	0	0	0	0	0	0
W							w1c	w1c	w1c							
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_HSM_FUNC	0	0	0	0	F_JTAG_FUNC	0	0	0	F_FCCU_SOFT	F_FCCU_HARD	0	F_SOFT_FUNC	F_ST_DONE	F_ESR1	0
W	w1c					w1c				w1c	w1c		w1c	w1c	w1c	
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1031. ‘Functional’ Event Status Register (RGM_FES)

Table 1057. RGM_FES field descriptions

Field	Description
6 F_VOR_FUNC_STDBY	Flag for voltage out of range from LVDs or HVDs, or both, during standby ‘functional’ reset 0 No voltage out of range ‘functional’ reset event has occurred since either the last clear or the last power-on reset assertion 1 A voltage out of range ‘functional’ reset event has occurred
7 F_VOR_FUNC	Flag for voltage out of range ‘functional’ reset 0 No voltage out of range ‘functional’ reset event has occurred since either the last clear or the last power-on reset assertion 1 A voltage out of range ‘functional’ reset event has occurred
8 F_TSR_FUNC	Flag for temperature sensor ‘functional’ reset 0 No temperature sensor 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A temperature sensor 'functional' reset event has occurred
16 F_HSM_FUNC	Flag for HSM ‘functional’ reset request 0 No HSM 'functional' reset request event has occurred since either the last clear or the last power-on reset assertion 1 A HSM 'functional' reset request event has occurred

Table 1057. RGM_FES field descriptions (continued)

Field	Description
21 F_JTAG_FUNC	Flag for JTAG ‘functional’ reset or stand-by exit 0 No JTAG ‘functional’ reset event has occurred since either the last clear or the last power-on reset assertion, or no exit from stand-by 1 A JTAG ‘functional’ reset event has occurred, or a PORST functional reset during the stand-by mode occurred
25 F_FCCU_SOFT	Flag for FCCU soft reaction 0 No FCCU soft reaction event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU soft reaction event has occurred
26 F_FCCU_HARD	Flag for FCCU hard reaction reset 0 No FCCU hard reaction reset event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU hard reaction reset event has occurred
28 F_SOFT_FUNC	Flag for software ‘functional’ reset 0 No software ‘functional’ reset event has occurred since either the last clear or the last power-on reset assertion 1 A software ‘functional’ reset event has occurred
29 F_ST_DONE	Flag for self test completed 0 No self test completed event has occurred since either the last clear or the last power-on reset assertion 1 A self test completed event has occurred
30 F_ESR1	Flag for ESR1 External Reset 0 No ESR1 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR1 external reset event has occurred

51.3.1.6 ‘Functional’ Event Reset Disable Register (RGM_FERD)

This register provides dedicated bits to disable ‘functional’ reset sources.When a ‘functional’ reset source is disabled, the associated ‘functional’ event will trigger either a SAFE mode request or an interrupt request (refer to *Section 51.3.1.7: ‘Functional’ Event Alternate Request Register (RGM_FEAR)*). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

This register is reset only on power-on and any ‘destructive’ reset.

Caution: It is important to clear the RGM_FES register before setting any of the bits in the RGM_FERD register to ‘1’. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0x310 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	D_VOR_FUNC_STDBY	D_VOR_FUNC	D_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	D_HSM_FUNC	0	0	0	0	D_JTAG_FUNC	0	0	0	D_FCCU_SOFT	D_FCCU_HARD	0	D_SOFT_FUNC	D_ST_DONE	D_ESR1	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1032. 'Functional' Event Reset Disable Register (RGM_FERD)

Table 1058. RGM_FERD field descriptions

Field	Description
6 D_VOR_FUNC_STDBY	Disable voltage out of range from LVDs or HVDs, or both, during standby 'functional' reset 0 A voltage out of range 'functional' reset event triggers a reset sequence 1 A voltage out of range 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC_STDBY
7 D_VOR_FUNC	Disable voltage out of range 'functional' reset 0 A voltage out of range 'functional' reset event triggers a reset sequence 1 A voltage out of range 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC
8 D_TSR_FUNC	Disable temperature sensor 'functional' reset 0 A temperature sensor 'functional' reset event triggers a reset sequence 1 A temperature sensor 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_TSR_FUNC
16 D_HSM_FUNC	Disable HSM 'functional' reset request 0 A HSM 'functional' reset request event triggers a reset sequence
21 D_JTAG_FUNC	Disable JTAG 'functional' reset 0 A JTAG 'functional' reset event triggers a reset sequence

Table 1058. RGM_FERD field descriptions (continued)

Field	Description
25 D_FCCU_SOFT	Disable FCCU soft reaction 0 A FCCU soft reaction event triggers a reset sequence
26 D_FCCU_HARD	Disable FCCU hard reaction reset 0 A FCCU hard reaction reset event triggers a reset sequence
28 D_SOFT_FUNC	Disable software ‘functional’ reset 0 A software ‘functional’ reset event triggers a reset sequence
29 D_ST_DONE	Disable self test completed 0 A self test completed event triggers a reset sequence
30 D_ESR1	Disable ESR1 External Reset 0 An ESR1 external reset event triggers a reset sequence 1 An ESR1 external reset event generates either a SAFE mode or an interrupt request depending on the values in RGM_FEAR.AR_ESR1

51.3.1.7 ‘Functional’ Event Alternate Request Register (RGM_FEAR)

This register defines an alternate request to be generated when a reset on a ‘functional’ event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0x320 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	AR_VOR_FUNC_STDBY	AR_VOR_FUNC	AR_TSR_FUNC	0	0	0	0	0	0	0
W																
‘Dest’ Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AR_ESR1	0
W																
‘Dest’ Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1033. ‘Functional’ Event Alternate Request Register (RGM_FEAR)

Table 1059. RGM_FEAR field descriptions

Field	Description
6 AR_VOR_FUNC_STDBY	Alternate Request for LVDs or HVDs, or both, during standby voltage out of range 'functional' reset 0 Generate a SAFE mode request on a voltage out of range 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a voltage out of range 'functional' reset event if the reset is disabled
7 AR_VOR_FUNC	Alternate Request for voltage out of range 'functional' reset 0 Generate a SAFE mode request on a voltage out of range 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a voltage out of range 'functional' reset event if the reset is disabled
8 AR_TSR_FUNC	Alternate Request for temperature sensor 'functional' reset 0 Generate a SAFE mode request on a temperature sensor 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor 'functional' reset event if the reset is disabled
30 AR_ESR1	Alternate Request for ESR1 External Reset 0 Generate a SAFE mode request on a ESR1 external reset event if the reset is disabled 1 Generate an interrupt request on a ESR1 external reset event if the reset is disabled

51.3.1.8 'Functional' Reset Enable Register (RGM_FBRE)

This register enables the generation of an external reset on 'functional' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0x330 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	BE_VOR_FUNC_STDBY	BE_VOR_FUNC	BE_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE_HSM_FUNC	0	0	0	0	BE_JTAG_FUNC	0	0	0	BE_FCCU_SOFT	BE_FCCU_HARD	0	BE_SOFT_FUNC	BE_ST_DONE	BE_ESR1	0
W																
'Dest' Reset	1	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0

Figure 1034. 'Functional' Reset Enable Register (RGM_FBRE)

Table 1060. RGM_FBRE field descriptions

Field	Description
6 BE_VOR_FUNC_STDBY	Reset Enable for LVDs or HVDs, or both, voltage out of range during standby 'functional' reset 0 ESR0 is asserted on a voltage out of range 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a voltage out of range 'functional' reset event
7 BE_VOR_FUNC	Reset Enable for voltage out of range 'functional' reset 0 ESR0 is asserted on a voltage out of range 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a voltage out of range 'functional' reset event
8 BE_TSR_FUNC	Reset Enable for temperature sensor 'functional' reset 0 ESR0 is asserted on a temperature sensor 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a temperature sensor 'functional' reset event
16 BE_HSM_FUNC	Reset Enable for HSM 'functional' reset request 0 ESR0 is asserted on a HSM 'functional' reset request event 1 ESR0 is not asserted on a HSM 'functional' reset request event
21 BE_JTAG_FUNC	Reset Enable for JTAG 'functional' reset 0 ESR0 is asserted on a JTAG 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a JTAG 'functional' reset event

Table 1060. RGM_FBRE field descriptions (continued)

Field	Description
25 BE_FCCU_SOFT	Reset Enable for FCCU soft reaction 0 ESR0 is asserted on a FCCU soft reaction event 1 ESR0 is not asserted on a FCCU soft reaction event
26 BE_FCCU_HARD	Reset Enable for a FCCU hard reaction 0 ESR0 is asserted on a FCCU hard reaction reset event 1 ESR0 is not asserted on a FCCU hard reaction reset event
28 BE_SOFT_FUNC	Reset Enable for software ‘functional’ reset 0 ESR0 is asserted on a software ‘functional’ reset event if the reset is enabled 1 ESR0 is not asserted on a software ‘functional’ reset event
29 BE_ST_DONE	Reset Enable for self test completed 0 ESR0 is asserted on a self test completed event
30 BE_ESR1	Reset Enable for ESR1 External Reset 0 ESR0 is asserted on an ESR1 external reset event if the reset is enabled 1 ESR0 is not asserted on an ESR1 external reset event

Note: For more details about ESR0, refer to Section 51.4.3.2: ESR0 Output External reset.

51.3.1.9 ‘Functional’ Event Short Sequence Register (RGM_FESS)

This register defines which reset sequence will be done when a ‘functional’ reset sequence is triggered. The ‘functional’ reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2.

Note: This could be useful for fast reset sequence, for example to skip flash memory reset.

It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0x340 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	SS_VOR_FUNC_STDBY	SS_VOR_FUNC	SS_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SS_HSM_FUNC	0	0	0	0	SS_JTAG_FUNC	0	0	0	SS_FCCU_SOFT	SS_FCCU_HARD	0	SS_SOFT_FUNC	SS_ST_DONE	SS_ESR1	0
W																
'Dest' Reset	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 1035. 'Functional' Event Short Sequence Register (RGM_FESS)

Table 1061. RGM_FESS field descriptions

Field	Description
6 SS_VOR_FUNC_STDBY	Short Sequence for LVDs or HVDs, or both, voltage out of range during standby 'functional' reset 0 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE3
7 SS_VOR_FUNC	Short Sequence for voltage out of range 'functional' reset 0 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE3
8 SS_TSR_FUNC	Short Sequence for temperature sensor 'functional' reset 0 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE3
16 SS_HSM_FUNC	Short Sequence for HSM 'functional' reset request 1 The reset sequence triggered by a HSM 'functional' reset request event will start from PHASE3

Table 1061. RGM_FESS field descriptions (continued)

Field	Description
21 SS_JTAG_FUNC	Short Sequence for JTAG ‘functional’ reset 0 The reset sequence triggered by a JTAG ‘functional’ reset event will start from PHASE1 1 The reset sequence triggered by a JTAG ‘functional’ reset event will start from PHASE3
25 SS_FCCU_SOFT	Short Sequence for FCCU soft reaction 1 The reset sequence triggered by a FCCU soft reaction event will start from PHASE3
26 SS_FCCU_HARD	Short Sequence for a FCCU hard reaction 0 The reset sequence triggered by a FCCU hard reaction reset event will start from PHASE1
28 SS_SOFT_FUNC	Short Sequence for software ‘functional’ reset 0 The reset sequence triggered by a software ‘functional’ reset event will start from PHASE1 1 The reset sequence triggered by a software ‘functional’ reset event will start from PHASE3
29 SS_ST_DONE	Short Sequence for self test completed 0 The reset sequence triggered by a self test completed event will start from PHASE1
30 SS_ESR1	Short Sequence for ESR1 External Reset 0 The reset sequence triggered by an ESR1 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR1 external reset event if the reset is enabled will start from PHASE3

51.3.1.10 ‘Functional’ Reset Escalation Threshold Register (RGM_FRET)

This register sets the threshold for ‘functional’ reset escalation to a ‘destructive’ reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the FRET field will enable the ‘functional’ reset escalation function. Writing any value to this register will reset the ‘functional’ reset counter. Refer to *Section 51.4.6: ‘Functional’ reset escalation* for details on the ‘functional’ reset escalation function.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0x604 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	FRET			
W								
‘Dest’ Reset	0	0	0	0	0	1	0	0

Figure 1036. ‘Functional’ Reset Escalation Threshold Register (RGM_FRET)

Table 1062. RGM_FRET field descriptions

Field	Description
4:7 FRET	'Functional' Reset Escalation Threshold 0000 'Functional' reset escalation function disabled 0001 Number of 'functional' reset events before escalation to destructive reset … 1111 Number of 'functional' reset events before escalation to destructive reset

51.3.1.11 'Destructive' Reset Escalation Threshold Register (RGM_DRET)

This register sets the threshold for 'destructive' reset escalation keeping the chip in the reset state until the next power-on reset triggers a new reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the DRET field will enable the 'destructive' reset software escalation function. Writing any value to this register will reset the 'destructive' reset counter. Refer to *Section 51.4.7: 'Destructive' reset escalation*, for details on the 'destructive' reset escalation function.

This register is reset only on power-on.

Address: 0x608 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	DRET			
W								
POR	0	0	0	0	1	0	0	0

Figure 1037. 'Destructive' Reset Escalation Threshold Register (RGM_DRET)

Table 1063. RGM_DRET field descriptions

Field	Description
4:7 DRET	'Destructive' Reset Escalation Threshold 0000 'Destructive' reset escalation function disabled 0001 Number of 'destructive' reset events before chip enters permanent reset state … 1111 Number of 'destructive' reset events before chip enters permanent reset state

51.3.1.12 Peripheral Reset Register 0 (RGM_PRST0)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Refer to *Table 1064* for details.

Address: 0x610 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_0_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIUL_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1038. Peripheral Reset Register 0 (RGM_PRST0)

Table 1064. RGM_PRST0 register field descriptions

Field	Description
1 PIT_0_RST	PIT_0 reset Writing a ‘1’ to this bit resets the PIT_0 peripheral. Writing a ‘0’ to this bit releases the reset to the PIT_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
16 SIUL_RST	SIUL reset Writing a ‘1’ to this bit resets the SIUL peripheral. Writing a ‘0’ to this bit releases the reset to the SIUL peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

51.3.1.13 Peripheral Reset Register 1 (RGM_PRST1)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Refer to *Table 1065* for details.

Address: 0x614 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CCCU_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CRC_0_RST	0	DMAMUX_0_RST				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1039. Peripheral Reset Register 1 (RGM_PRST1)

Table 1065. RGM_PRST1 register field descriptions

Field	Description
1 CCCU_RST	CCCU reset Writing a ‘1’ to this bit resets the CCCU peripheral. Writing a ‘0’ to this bit releases the reset to the CCCU peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
25 CRC_0_RST	CRC_0 reset Writing a ‘1’ to this bit resets the CRC_0 peripheral. Writing a ‘0’ to this bit releases the reset to the CRC_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
27 DMAMUX_0_RST	DMAMUX_0 reset Writing a ‘1’ to this bit resets the DMAMUX_0 peripheral. Writing a ‘0’ to this bit releases the reset to the DMAMUX_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

51.3.1.14 Peripheral Reset Register 2 (RGM_PRST2)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Refer to *Table 1066* for details.

Address: 0x618 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFLEXD_0_RST	LINFLEXD_2_RST	LINFLEXD_4_RST	LINFLEXD_6_RST	LINFLEXD_8_RST	LINFLEXD_10_RST	LINFLEXD_12_RST	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_SUB_0_CTR_RST	0	M_CAN_0_SUB_0_RST	0	0	M_CAN_1_SUB_0_RST	M_CAN_2_SUB_0_RST	M_CAN_3_SUB_0_RST	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1040. Peripheral Reset Register 2 (RGM_PRST2)

Table 1066. RGM_PRST2 register field descriptions

Field	Description
3 LINFLEXD_0_RST	LINFLEXD_0 reset Writing a ‘1’ to this bit resets the LINFLEXD_0 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 LINFLEXD_2_RST	LINFLEXD_2 reset Writing a ‘1’ to this bit resets the LINFLEXD_2 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
5 LINFLEXD_4_RST	LINFLEXD_4 reset Writing a ‘1’ to this bit resets the LINFLEXD_4 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_4 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1066. RGM_PRST2 register field descriptions (continued)

Field	Description
6 LINFLEXD_6_RST	LINFLEXD_6 reset Writing a ‘1’ to this bit resets the LINFLEXD_6 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_6 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
7 LINFLEXD_8_RST	LINFLEXD_8 reset Writing a ‘1’ to this bit resets the LINFLEXD_8 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_8 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
8 LINFLEXD_10_RST	LINFLEXD_10 reset Writing a ‘1’ to this bit resets the LINFLEXD_10 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_10 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
9 LINFLEXD_12_RST	LINFLEXD_12 reset Writing a ‘1’ to this bit resets the LINFLEXD_12 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_12 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
21 CAN_RAM_SUB_0_CTR_RST	CAN_RAM_SUB_0_CTR reset Writing a ‘1’ to this bit resets the CAN_RAM_SUB_0_CTR peripheral. Writing a ‘0’ to this bit releases the reset to the CAN_RAM_SUB_0_CTR peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral **Note:** RGM is able to reset CAN RAM controller and not CAN RAM content. In other words, it is not possible to access CAN RAM content if corresponding CAN RAM controller is kept under reset by CAN_RAM_SUB_0_CTR_RST bit.
23 M_CAN_0_SUB_0_RST	M_CAN_0_SUB_0 reset Writing a ‘1’ to this bit resets the M_CAN_0_SUB_0 peripheral. Writing a ‘0’ to this bit releases the reset to the M_CAN_0_SUB_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1066. RGM_PRST2 register field descriptions (continued)

Field	Description
26 M_CAN_1_SUB_0_RST	M_CAN_1_SUB_0 reset Writing a ‘1’ to this bit resets the M_CAN_1_SUB_0 peripheral. Writing a ‘0’ to this bit releases the reset to the M_CAN_1_SUB_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
27 M_CAN_2_SUB_0_RST	M_CAN_2_SUB_0 reset Writing a ‘1’ to this bit resets the M_CAN_2_SUB_0 peripheral. Writing a ‘0’ to this bit releases the reset to the M_CAN_2_SUB_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 M_CAN_3_SUB_0_RST	M_CAN_3_SUB_0 reset Writing a ‘1’ to this bit resets the M_CAN_3_SUB_0 peripheral. Writing a ‘0’ to this bit releases the reset to the M_CAN_3_SUB_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

51.3.1.15 Peripheral Reset Register 3 (RGM_PRST3)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Refer to *Table 1067* for details.

Address: 0x61C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAR_ADC_12BIT_0_RST	0	0	0	0	0	0	0	0	0	0	0	0	SAR_ADC_10BIT_STDBY_RST	0	SAR_ADC_12BIT_B_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	IIC_0_RST	0	DSPI_0_RST	DSPI_2_RST	DSPI_4_RST	DSPI_6_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1041. Peripheral Reset Register 3 (RGM_PRST3)

Table 1067. RGM_PRST3 register field descriptions

Field	Description
0 SAR_ADC_12BIT_0_RST	SAR_ADC_12BIT_0 reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_0 peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
13 SAR_ADC_10BIT_ STDBY_RST	SAR_ADC_10BIT_STDBY reset Writing a ‘1’ to this bit resets the SAR_ADC_10BIT_STDBY peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_10BIT_STDBY peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
15 SAR_ADC_12BIT_B_RST	SAR_ADC_12BIT_B reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_B peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_B peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral **Note:** In case BCTU_0 is used to trigger SAR_ADC_12BIT_B conversions, force the reset of BCTU_0 before setting SAR_ADC_12BIT_B_RST to 1'b1 by RGM_PRST7[BODY_CTU_0_RST] bit. De-assert BCTU_0 reset (clearing BODY_CTU_0_RST to 1’b0) only after SAR_ADC_12BIT_B is out of reset (clearing SAR_ADC_12BIT_B_RST to 1'b0).
26 IIC_0_RST	IIC_0 reset Writing a ‘1’ to this bit resets the IIC_0 peripheral. Writing a ‘0’ to this bit releases the reset to the IIC_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 DSPI_0_RST	DSPI_0 reset Writing a ‘1’ to this bit resets the DSPI_0 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
29 DSPI_2_RST	DSPI_2 reset Writing a ‘1’ to this bit resets the DSPI_2 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1067. RGM_PRST3 register field descriptions (continued)

Field	Description
30 DSPI_4_RST	DSPI_4 reset Writing a ‘1’ to this bit resets the DSPI_4 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_4 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
31 DSPI_6_RST	DSPI_6 reset Writing a ‘1’ to this bit resets the DSPI_6 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_6 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

51.3.1.16 Peripheral Reset Register 4 (RGM_PRST4)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Refer to *Table 1068* for details.

Address: 0x620 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_1_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAR_ADC_12BIT_0_SEQ_RST	0	SAR_ADC_12BIT_1_SEQ_RST	0	0	0	0	0	0	0	0	0	0	SAR_ADC_10BIT_STDBY_SEQ_RST	SAR_ADC_12BIT_B_SEQ_RST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1042. Peripheral Reset Register 4 (RGM_PRST4)

Table 1068. RGM_PRST4 register field descriptions

Field	Description
1 PIT_1_RST	PIT_1 reset Writing a '1' to this bit resets the PIT_1 peripheral. Writing a '0' to this bit releases the reset to the PIT_1 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
16 SAR_ADC_12BIT_0_SEQ _RST	SAR_ADC_12BIT_0_SEQ reset Writing a '1' to this bit resets the SAR_ADC_12BIT_0_SEQ peripheral. Writing a '0' to this bit releases the reset to the SAR_ADC_12BIT_0_SEQ peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
18 SAR_ADC_12BIT_1_SEQ _RST	SAR_ADC_12BIT_1_SEQ reset Writing a '1' to this bit resets the SAR_ADC_12BIT_1_SEQ peripheral. Writing a '0' to this bit releases the reset to the SAR_ADC_12BIT_1_SEQ peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
29 SAR_ADC_10BIT_STDBY _SEQ_RST	SAR_ADC_10BIT_STDBY_SEQ reset Writing a '1' to this bit resets the SAR_ADC_10BIT_STDBY_SEQ peripheral. Writing a '0' to this bit releases the reset to the SAR_ADC_10BIT_STDBY_SEQ peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
30 SAR_ADC_12BIT_B _SEQ_RST	SAR_ADC_12BIT_B_SEQ reset Writing a '1' to this bit resets the SAR_ADC_12BIT_B_SEQ peripheral. Writing a '0' to this bit releases the reset to the SAR_ADC_12BIT_B_SEQ peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

51.3.1.17 Peripheral Reset Register 5 (RGM_PRST5)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Refer to *Table 1069* for details.

Address: 0x624 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FCCU_RST	0	0	CRC_1_RST	0	DMAMUX_1_RST	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1043. Peripheral Reset Register 5 (RGM_PRST5)

Table 1069. RGM_PRST5 register field descriptions

Field	Description
22 FCCU_RST	FCCU reset Writing a ‘1’ to this bit resets the FCCU peripheral. Writing a ‘0’ to this bit releases the reset to the FCCU peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
25 CRC_1_RST	CRC_1 reset Writing a ‘1’ to this bit resets the CRC_1 peripheral. Writing a ‘0’ to this bit releases the reset to the CRC_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
27 DMAMUX_1_RST	DMAMUX_1 reset Writing a ‘1’ to this bit resets the DMAMUX_1 peripheral. Writing a ‘0’ to this bit releases the reset to the DMAMUX_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

51.3.1.18 Peripheral Reset Register 6 (RGM_PRST6)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Refer to *Table 1070* for details.

Address: 0x628 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFLEXD_1_RST	LINFLEXD_3_RST	LINFLEXD_5_RST	LINFLEXD_7_RST	LINFLEXD_9_RST	LINFLEXD_11_RST	0	LINFLEXD_15_RST	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_SUB_1_CTR_RST	0	0	0	M_CAN_1_SUB_1_RST	M_CAN_2_SUB_1_RST	M_CAN_3_SUB_1_RST	M_CAN_4_SUB_1_RST	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1044. Peripheral Reset Register 6 (RGM_PRST6)

Table 1070. RGM_PRST6 register field descriptions

Field	Description
3 LINFLEXD_1_RST	LINFLEXD_1 reset Writing a ‘1’ to this bit resets the LINFLEXD_1 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 LINFLEXD_3_RST	LINFLEXD_3 reset Writing a ‘1’ to this bit resets the LINFLEXD_3 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
5 LINFLEXD_5_RST	LINFLEXD_5 reset Writing a ‘1’ to this bit resets the LINFLEXD_5 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_5 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1070. RGM_PRST6 register field descriptions (continued)

Field	Description
6 LINFLEXD_7_RST	LINFLEXD_7 reset Writing a ‘1’ to this bit resets the LINFLEXD_7 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_7 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
7 LINFLEXD_9_RST	LINFLEXD_9 reset Writing a ‘1’ to this bit resets the LINFLEXD_9 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_9 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
8 LINFLEXD_11_RST	LINFLEXD_11 reset Writing a ‘1’ to this bit resets the LINFLEXD_11 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_11 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
10 LINFLEXD_15_RST	LINFLEXD_15 reset Writing a ‘1’ to this bit resets the LINFLEXD_15 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_15 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
21 CAN_RAM_SUB_1_ CTR_RST	CAN_RAM_SUB_1_CTR reset Writing a ‘1’ to this bit resets the CAN_RAM_SUB_1_CTR peripheral. Writing a ‘0’ to this bit releases the reset to the CAN_RAM_SUB_1_CTR peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral **Note:** RGM is able to reset CAN RAM controller and not CAN RAM content. In other words, it is not possible to access CAN RAM content if corresponding CAN RAM controller is kept under reset by CAN_RAM_SUB_0_CTR_RST bit.
25 M_CAN_1_SUB_1_RST	M_CAN_1_SUB_1 reset Writing a ‘1’ to this bit resets the M_CAN_1_SUB_1 peripheral. Writing a ‘0’ to this bit releases the reset to the M_CAN_1_SUB_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
26 M_CAN_2_SUB_1_RST	M_CAN_2_SUB_1 reset Writing a ‘1’ to this bit resets the M_CAN_2_SUB_1 peripheral. Writing a ‘0’ to this bit releases the reset to the M_CAN_1_SUB_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1070. RGM_PRST6 register field descriptions (continued)

Field	Description
27 M_CAN_3_SUB_1_RST	M_CAN_3_SUB_1 reset Writing a ‘1’ to this bit resets the M_CAN_3_SUB_1 peripheral. Writing a ‘0’ to this bit releases the reset to the M_CAN_3_SUB_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 M_CAN_4_SUB_1_RST	M_CAN_4_SUB_1 reset Writing a ‘1’ to this bit resets the M_CAN_4_SUB_1 peripheral. Writing a ‘0’ to this bit releases the reset to the M_CAN_4_SUB_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

51.3.1.19 Peripheral Reset Register 7 (RGM_PRST7)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Refer to *Table 1071* for details.

Address: 0x62C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAR_ADC_12BIT_1_RST	0	0	0	0	0	0	0	0	0	0	0	BODY_CTU_0_RST	STDBY_CTU_0_RST	EMIOS_0_RST	EMIOS_1_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	ETHERNET_0_RST	0	0	0	0	0	0	DSPI_1_RST	DSPI_3_RST	DSPI_5_RST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1045. Peripheral Reset Register 7 (RGM_PRST7)

Table 1071. RGM_PRST7 register field descriptions

Field	Description
0 SAR_ADC_12BIT_1_RST	SAR_ADC_12BIT_1 reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_1 peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
12 BODY_CTU_0_RST	BODY_CTU_0 reset Writing a ‘1’ to this bit resets the BODY_CTU_0 peripheral. Writing a ‘0’ to this bit releases the reset to the BODY_CTU_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
13 STDBY_CTU_0_RST	STDBY_CTU_0 reset Writing a ‘1’ to this bit resets the STDBY_CTU_0 peripheral. Writing a ‘0’ to this bit releases the reset to the STDBY_CTU_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
14 EMIOS_0_RST	EMIOS_0 reset Writing a ‘1’ to this bit resets the EMIOS_0 peripheral. Writing a ‘0’ to this bit releases the reset to the EMIOS_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
15 EMIOS_1_RST	EMIOS_1 reset Writing a ‘1’ to this bit resets the EMIOS_1 peripheral. Writing a ‘0’ to this bit releases the reset to the EMIOS_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
21 ETHERNET_0_RST	ETHERNET_0 reset Writing a ‘1’ to this bit resets the ETHERNET_0 peripheral. Writing a ‘0’ to this bit releases the reset to the ETHERNET_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 DSPI_1_RST	DSPI_1 reset Writing a ‘1’ to this bit resets the DSPI_1 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1071. RGM_PRST7 register field descriptions (continued)

Field	Description
29 DSPI_3_RST	DSPI_3 reset Writing a ‘1’ to this bit resets the DSPI_3 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
30 DSPI_5_RST	DSPI_5 reset Writing a ‘1’ to this bit resets the DSPI_5 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_5 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Caution: Use the RGM_PRST*n* registers with care. Refer to *Section 51.4.8: Individual peripheral resets* for the proper sequence to prevent unexpected chip behavior.

51.3.1.20 Peripheral Reset Status Register 0 (RGM_PSTAT0)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. Refer to *Table 1072* for details.

Address: 0x630 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_0_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIUL_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1046. Peripheral Reset Status Register 0 (RGM_PSTAT0)

Table 1072. RGM_PSTAT0 register field descriptions

Field	Description
1 PIT_0_STAT	PIT_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
16 SIUL_STAT	SIUL reset status 0 peripheral is not in reset 1 peripheral is in reset

51.3.1.21 Peripheral Reset Status Register 1 (RGM_PSTAT1)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. Refer to *Table 1073* for details.

Address: 0x634 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CCCU_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CRC_0_STAT	0	DMAMUX_0_STAT	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1047. Peripheral Reset Status Register 1 (RGM_PSTAT1)

Table 1073. RGM_PSTAT1 register field descriptions

Field	Description
1 CCCU_STAT	CCCU reset status 0 peripheral is not in reset 1 peripheral is in reset
25 CRC_0_STAT	CRC_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
27 DMAMUX_0_STAT	DMAMUX_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

51.3.1.22 Peripheral Reset Status Register 2 (RGM_PSTAT2)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. Refer to *Table 1074* for details.

Address: 0x638 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFLEXD_0_STAT	LINFLEXD_2_STAT	LINFLEXD_4_STAT	LINFLEXD_6_STAT	LINFLEXD_8_STAT	LINFLEXD_10_STAT	LINFLEXD_12_STAT	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_SUB_0_CTR_STAT	0	M_CAN_0_SUB_0_STAT	0	0	M_CAN_1_SUB_0_STAT	M_CAN_2_SUB_0_STAT	M_CAN_3_SUB_0_STAT	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1048. Peripheral Reset Status Register 2 (RGM_PSTAT2)

Table 1074. RGM_PSTAT2 register field descriptions

Field	Description
3 LINFLEXD_0_STAT	LINFLEXD_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 LINFLEXD_2_STAT	LINFLEXD_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
5 LINFLEXD_4_STAT	LINFLEXD_4 reset status 0 peripheral is not in reset 1 peripheral is in reset
6 LINFLEXD_6_STAT	LINFLEXD_6 reset status 0 peripheral is not in reset 1 peripheral is in reset
7 LINFLEXD_8_STAT	LINFLEXD_8 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1074. RGM_PSTAT2 register field descriptions (continued)

Field	Description
8 LINFLEXD_10_STAT	LINFLEXD_10 reset status 0 peripheral is not in reset 1 peripheral is in reset
9 LINFLEXD_12_STAT	LINFLEXD_12 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 CAN_RAM_SUB_0_ CTR_STAT	CAN_RAM_SUB_0_CTR reset status 0 peripheral is not in reset 1 peripheral is in reset
23 M_CAN_0_SUB_0_STAT	M_CAN_0_SUB_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
26 M_CAN_1_SUB_0_STAT	M_CAN_1_SUB_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
27 M_CAN_2_SUB_0_STAT	M_CAN_2_SUB_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 M_CAN_3_SUB_0_STAT	M_CAN_3_SUB_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

51.3.1.23 Peripheral Reset Status Register 3 (RGM_PSTAT3)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. Refer to *Table 1075* for details.

Address: 0x63C Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAR_ADC_12BIT_0_STAT	0	0	0	0	0	0	0	0	0	0	0	0	SAR_ADC_10BIT_STDBY_STAT	0	SAR_ADC_12BIT_B_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	IIC_0_STAT	0	DSPI_0_STAT	DSPI_2_STAT	DSPI_4_STAT	DSPI_6_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1049. Peripheral Reset Status Register 3 (RGM_PSTAT3)

Table 1075. RGM_PSTAT3 register field descriptions

Field	Description
0 SAR_ADC_12BIT_0_STAT	SAR_ADC_12BIT_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
13 SAR_ADC_10BIT_STDBY_STAT	SAR_ADC_10BIT_STDBY reset status 0 peripheral is not in reset 1 peripheral is in reset
15 SAR_ADC_12BIT_B_STAT	SAR_ADC_12BIT_B reset status 0 peripheral is not in reset 1 peripheral is in reset
26 IIC_0_STAT	IIC_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 DSPI_0_STAT	DSPI_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
29 DSPI_2_STAT	DSPI_2 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1075. RGM_PSTAT3 register field descriptions (continued)

Field	Description
30 DSPI_4_STAT	DSPI_4 reset status 0 peripheral is not in reset 1 peripheral is in reset
31 DSPI_6_STAT	DSPI_6 reset status 0 peripheral is not in reset 1 peripheral is in reset

51.3.1.24 Peripheral Reset Status Register 4 (RGM_PSTAT4)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. Refer to *Table 1076* for details.

Address: 0x640 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_1_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAR_ADC_12BIT_0_SEQ_STAT	0	SAR_ADC_12BIT_1_SEQ_STAT	0	0	0	0	0	0	0	0	0	0	SAR_ADC_10BIT_STDBY_SEQ_STAT	SAR_ADC_12BIT_B_SEQ_STAT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1050. Peripheral Reset Status Register 4 (RGM_PSTAT4)

Table 1076. RGM_PSTAT4 register field descriptions

Field	Description
1 PIT_1_STAT	PIT_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
16 SAR_ADC_12BIT_0_ SEQ_STAT	SAR_ADC_12BIT_0_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1076. RGM_PSTAT4 register field descriptions (continued)

Field	Description
18 SAR_ADC_12BIT_1_SEQ_STAT	SAR_ADC_12BIT_1_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
29 SAR_ADC_10BIT_STDBY_SEQ_STAT	SAR_ADC_10BIT_STDBY_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
30 SAR_ADC_12BIT_B_SEQ_STAT	SAR_ADC_12BIT_B_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset

51.3.1.25 Peripheral Reset Status Register 5 (RGM_PSTAT5)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. Refer to *Table 1077* for details.

Address: 0x644 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FCCU_STAT	0	0	CRC_1_STAT	0	DMAMUX_1_STAT	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1051. Peripheral Reset Status Register 5 (RGM_PSTAT5)

Table 1077. RGM_PSTAT5 register field descriptions

Field	Description
22 FCCU_STAT	FCCU reset status 0 peripheral is not in reset 1 peripheral is in reset
25 CRC_1_STAT	CRC_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
27 DMAMUX_1_STAT	DMAMUX_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

51.3.1.26 Peripheral Reset Status Register 6 (RGM_PSTAT6)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. Refer to *Table 1078* for details.

Address: 0x648 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFLEXD_1_STAT	LINFLEXD_3_STAT	LINFLEXD_5_STAT	LINFLEXD_7_STAT	LINFLEXD_9_STAT	LINFLEXD_11_STAT	0	LINFLEXD_15_STAT	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_SUB_1_CTR_STAT	0	0	0	M_CAN_1_SUB_1_STAT	M_CAN_2_SUB_1_STAT	M_CAN_3_SUB_1_STAT	M_CAN_4_SUB_1_STAT	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1052. Peripheral Reset Status Register 6 (RGM_PSTAT6)

Table 1078. RGM_PSTAT6 register field descriptions

Field	Description
3 LINFLEXD_1_STAT	LINFLEXD_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 LINFLEXD_3_STAT	LINFLEXD_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
5 LINFLEXD_5_STAT	LINFLEXD_5 reset status 0 peripheral is not in reset 1 peripheral is in reset
6 LINFLEXD_7_STAT	LINFLEXD_7 reset status 0 peripheral is not in reset 1 peripheral is in reset
7 LINFLEXD_9_STAT	LINFLEXD_9 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1078. RGM_PSTAT6 register field descriptions (continued)

Field	Description
8 LINFLEXD_11_STAT	LINFLEXD_11 reset status 0 peripheral is not in reset 1 peripheral is in reset
10 LINFLEXD_15_STAT	LINFLEXD_15 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 CAN_RAM_SUB_1_ CTR_STAT	CAN_RAM_SUB_1_CTR reset status 0 peripheral is not in reset 1 peripheral is in reset
25 M_CAN_1_SUB_1_STAT	M_CAN_1_SUB_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
26 M_CAN_2_SUB_1_STAT	M_CAN_2_SUB_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
27 M_CAN_3_SUB_1_STAT	M_CAN_3_SUB_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 M_CAN_4_SUB_1_STAT	M_CAN_4_SUB_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

51.3.1.27 Peripheral Reset Status Register 7 (RGM_PSTAT7)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. Refer to *Table 1079* for details.

Address: 0x64C Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAR_ADC_12BIT_1_STAT	0	0	0	0	0	0	0	0	0	0	0	BODY_CTU_0_STAT	STDBY_CTU_0_STAT	EMIOS_0_STAT	EMIOS_1_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	ETHERNET_0_STAT	0	0	0	0	0	0	DSPI_1_STAT	DSPI_3_STAT	DSPI_5_STAT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1053. Peripheral Reset Status Register 7 (RGM_PSTAT7)

Table 1079. RGM_PSTAT7 register field descriptions

Field	Description
0 SAR_ADC_12BIT_1_STAT	SAR_ADC_12BIT_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
12 BODY_CTU_0_STAT	BODY_CTU_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
13 STDBY_CTU_0_STAT	STDBY_CTU_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
14 EMIOS_0_STAT	EMIOS_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
15 EMIOS_1_STAT	EMIOS_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 ETHERNET_0_STAT	ETHERNET_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1079. RGM_PSTAT7 register field descriptions (continued)

Field	Description
28 DSPI_1_STAT	DSPI_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
29 DSPI_3_STAT	DSPI_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
30 DSPI_5_STAT	DSPI_5 reset status 0 peripheral is not in reset 1 peripheral is in reset

51.4 Functional description

51.4.1 Reset state machine

The main role of the MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in *Table 1080*.

Table 1080. MC_RGM reset implications

Source	What Gets Reset	External Reset Assertion(1)
power-on reset	all executes a start-up self test if enabled	yes
‘destructive’ resets	all except some clock/reset management executes a start-up self test if enabled	yes
long external reset	all except some clock/reset management and debug executes a start-up self test if enabled	programmable(2)
long external and long 'functional' resets	all except some clock/reset management, STCU, FCCU, and debug	programmable(2)
shortened ‘functional’ resets(3)	flip-flops except some clock/reset management , STCU, FCCU, and debug	programmable(2)

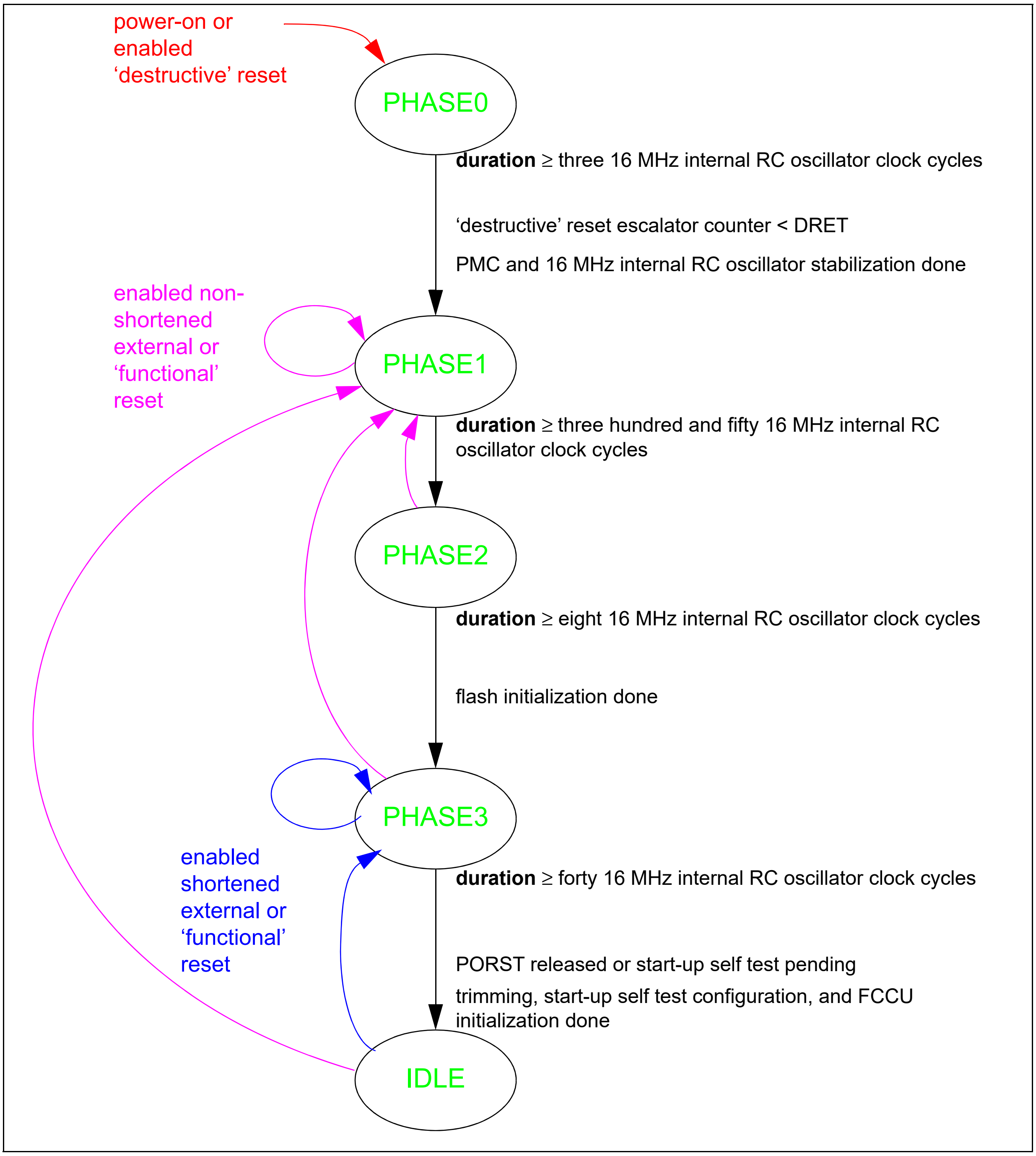
1. ‘external reset assertion’ means that the ESR0 pin is asserted by the MC_RGM from the start of the reset sequence until the end of reset PHASE3.
2. the assertion of the external reset is controlled via the RGM_FBRE register.
3. the short sequence is enabled via the RGM_FESS register.

Note: JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in *Figure 1054*.

Figure 1054. MC_RGM state machine



51.4.1.1 PHASE0 Phase

This phase is entered immediately from any phase on a power-on or enabled ‘destructive’ reset event. The reset state machine exits PHASE0 and enters PHASE1 on verification of the following:

- all enabled ‘destructive’ resets have been processed
- all processes that need to be done in PHASE0 are completed
  - PMC and 16 MHz internal RC oscillator stabilization
- a minimum of three 16 MHz internal RC oscillator clock cycles have elapsed since power-up completion and the last enabled ‘destructive’ reset event
- the ‘destructive’ reset escalator counter has not reached the value in the RGM_DRET[DRET] field

51.4.1.2 PHASE1 phase

This phase is entered either on exit from PHASE0 or immediately from PHASE2, PHASE3, or IDLE on a non-masked external or ‘functional’ reset event if it has not been configured to trigger a ‘short’ sequence. The reset state machine exits PHASE1 and enters PHASE2 on verification of the following:

- all enabled, non-shortened ‘functional’ resets have been processed
- a minimum of three hundred and fifty 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened ‘functional’ reset event

51.4.1.3 PHASE2 phase

This phase is entered on exit from PHASE1. The reset state machine exits PHASE2 and enters PHASE3 on verification of the following:

- all processes that need to be done in PHASE2 are completed
  - flash initialization
- a minimum of eight 16 MHz internal RC oscillator clock cycles have elapsed since entering PHASE2

51.4.1.4 PHASE3 phase

This phase is entered either on exit from PHASE2 or immediately from IDLE on an enabled, shortened ‘functional’ reset event. The reset state machine exits PHASE3 and enters IDLE on verification of the following:

- all processes that need to be done in PHASE3 are completed
  - trimming, start-up self test configuration, and FCCU initialization
- a minimum of forty 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled, shortened ‘functional’ reset event
- PORST is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence

51.4.1.5 IDLE phase

This is the final phase and is entered on exit from PHASE3. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

51.4.2 'Destructive' resets

A 'destructive' reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed.

The status flag associated with a given 'destructive' reset event (RGM_DES.F_<destructive reset> bit) is set when the 'destructive' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'destructive' reset can be optionally disabled by writing bit RGM_DERD.D_<destructive reset>.

Note: The RGM_DERD register can be written only once between two 'destructive' reset events.

The chip's low-voltage detector threshold ensures that, when 'destructive' voltage out of range (LVD or HVD, or both) is enabled, the supply is sufficient to have the 'destructive' event correctly propagated through the digital logic. Therefore, if a given 'destructive' reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given 'destructive' reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled 'destructive' reset will trigger a reset sequence starting from the beginning of PHASE0.

51.4.2.1 External power-on reset: PORST

The bidirectional external power-on reset pin PORST is a special case 'destructive' reset.

The detection of a falling edge on PORST will start the reset sequence.

The chip asserts PORST if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event

In this case, PORST is asserted until the end of PHASE0.

The PORST input is disabled immediately as of the PORST output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 us after the PORST output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

In addition, the reset sequence is not exited into application until PORST is no longer driven low from off-chip. In the case the chip is configured to execute a self-test on power-on, 'destructive', and external resets, this is the exit of PHASE3 after self test execution has completed.

51.4.3 External reset

51.4.3.1 ESR1 Input External reset

PA[4] can be configured as ESR1 input external reset through WKPU registers (refer to *Figure 17: NMI block diagram SPC584Bx*).

The status flag associated with the external reset falling edge event (RGM_FES.F_ESR1 bit) is set when the external reset is asserted and the power-on reset is not asserted.

Note: The RGM_FERD register can be written only once between two 'destructive' reset events.

Reset Generation Module (MC_RGM): An enabled external reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by the external reset. When RGM_FESS.SS_ESR1 is set, the external reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially when an external reset should not reset the Flash memory.

51.4.3.2 ESR0 Output External reset

PC[1] can be configured as ESR0 output external reset by programming the DCF Record ESR0_CONFIG.

The MC_RGM may assert the external reset ESR0 if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event
- an ESR1 external reset event if configured via the RGM_FBRE register to assert the external reset ESR0
- a 'functional' reset event configured via the RGM_FBRE register to assert the external reset ESR0

In this case, ESR0 is asserted until all conditions for exiting PHASE3 have been met.

51.4.4 ‘Functional’ resets

A ‘functional’ reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content are still intact.

The status flag associated with a given ‘functional’ reset event (RGM_FES.F_<functional reset> bit) is set when the ‘functional’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

A given ‘functional’ reset can be optionally disabled by software writing bit RGM_FERD.D_<functional reset>.

Note: The RGM_FERD register can be written only once between two power-on reset events.

An enabled ‘functional’ reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by a ‘functional’ reset. When RGM_FESS.SS_<functional reset> is set, the associated ‘functional’ reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially in case a ‘functional’ reset should not reset the flash memory module.

Refer to *Chapter 56: Mode Entry Module (MC_ME)*, for details on the STANDBY0 and DRUN modes.

51.4.5 Alternate event generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the RGM_DERD/FERD and RGM_DEAR/FEAR registers as shown in *Table 1081*.

Table 1081. MC_RGM alternate event selection

RGM_DERD/FERD bit value	RGM_DEAR/FEAR bit value	Generated event
0	X	reset
1	0	SAFE mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (that is at the reset source that caused the alternate request) and also clearing the appropriate RGM_DES/FES status bit.

Note: SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock, and therefore, if the system clock was disabled at the time of the event, the interrupt request is not asserted until the system clock is re-enabled.

If a masked 'destructive' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE0, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

If a masked 'functional' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE1, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

51.4.6 'Functional' reset escalation

'Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' or external resets has occurred between software writes to the RGM_FRET register. This function is enabled by writing a non-zero value to the FRET field of this register.

Once 'functional' reset escalation has been enabled, the MC_RGM increases a counter on each 'functional' or external reset which causes a reset sequence to be initiated (that is entrance into PHASE1 or PHASE3 from the IDLE phase). This counter is cleared on a write of any value to the RGM_FRET register and on any power-on or 'destructive' reset. If the counter reaches the value in the RGM_FRET register's FRET field, the MC_RGM starts a reset sequence at PHASE0 and asserts the F_EDR bit in the RGM_DES register.

51.4.7 'Destructive' reset escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until a new power-on triggers a reset sequence if a number of 'destructive' resets has occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

Once 'destructive' reset escalation has been enabled, the MC_RGM increases a counter on each 'destructive' reset[s] which causes a reset sequence to be initiated (that is entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (that is entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the value in the RGM_DRET register's DRET field, the MC_RGM enters reset PHASE0 and stays there until the next power-on reset occurs.

51.4.8 Individual peripheral resets

A peripheral may be reset individually without resetting the rest of the chip by setting the appropriate bit in the RGM_PRST*n* registers. However, in order to prevent unexpected behavior by the chip, the following sequence must be observed by software.

1. Disable the peripheral that is to be reset via the MC_ME (Refer to the MC_ME chapter for details).
2. Wait for the mode change to complete.
3. Set the bit corresponding to the peripheral in the RGM_PRST*n* registers.
4. Wait until the bit corresponding to the peripheral in the RGM_PSTAT*n* registers is set.
5. Clear the bit corresponding to the peripheral in the RGM_PRST*n* registers.
6. Wait until the bit corresponding to the peripheral in the RGM_PSTAT*n* registers is cleared.

The peripheral can be kept in reset indefinitely by not executing steps *5* and *6*. After the completion of step *6*, the peripheral can be enabled again via the MC_ME.

s. Only exception is the external power-on reset pin PORST that does not affect the increase of counter.

52 Boot Assist Flash (BAF)

52.1 Introduction

The BAF code is programmed by ST. The two main tasks of BAF are to provide a serial bootloader feature and to search for the application entry point.
For details regarding how the BAF code is executed, refer to *Chapter 8: Reset and Boot*.

The BAF is executed via the IRCOSC as clock source.

The MCU is booted through a collaboration of several blocks, hardware and firmware. The first boot phases are performed by a state machine inside the System Status and Configuration Module (SSCM). Once completed, the SSCM sends a reset vector to the HW boot core of the device pointing into the Boot Assist Flash (BAF).

The BAF code then checks the life cycle of the device. When the Life Cycle is set to FAIL_ANALYSIS, the BAF code shall check the UOPS[BAF_FA] bit inside SSCM as described in *Chapter 53: System Status and Configuration Module (SSCM)*. This bit mirrors the bit 31 of the UTEST_Miscellaneous_2 DCF. It tells whether the BAF shall stay in an endless loop or not when the Life Cycle is set to FAIL_ANALYSIS. In case the DCF is not programmed, the bit shall be 0 (refer to *Figure 1055*). Otherwise, it searches for a boot header and boots the application code in internal flash memory.

If no boot header is found in internal flash memory, it downloads application code serially using the LINFlexD module or the M_CAN module. The package pins used by LINFlexDs are the same pins used by the M_CAN modules. Hence, the external PHY can be either LIN or CAN (refer to *Section 52.3.8* for configuration details).

52.2 Memory map

For the BAF location in flash, refer to the Embedded memories chapter. The BAF block base address is 0x0040_4000. It is one time programmable (OTP) and is programmed during factory test. The address space and memory used by the BAF code are shown in *Table 1082*.

Note: The BAF Flash block is assigned as OTP and it is protected from programming by a bit in the PASS LOCK0 registers; but the BAF Flash block can further be protected from programming by an external hardware pin. If the Hardware Interlock feature is enabled, while this pin is logic ‘0’ the BAF Flash block cannot be programmed.

Table 1082. BAF memory organization

BAF Component	Address
BAF image header	0x0040_4000
BAF code entry point	0x0040_4100
BAF data area and stack	0x5280_0000–0x5280_0603

52.2.1 BAF image header

BAF image contains a 256-byte header starting from address 0x0040_4000. The BAF header is explained in *Table 1083*.

Table 1083. BAF image header

Address offset	Size (bits)	Field description
0x0000	32	SSCM boot record tag
0x0004	32	BAF start address
0x0008	32	BAF version number
0x000C–0x00FF	Reserved	

52.2.1.1 BAF image version

The BAF version is a 32-bit field in the image header starting at address 0x0040_4000. The 32-bit field is explained in *Table 1084*.

Table 1084. BAF image version

Field	Description
0:7	Major Number: This field contains the major version number for the BAF image
8:15	Minor Number: This field contains the minor version number for the BAF image
16:31	Reserved

52.3 Functional description

A functional description of the BAF modes is contained in the following sections.

52.3.1 Boot modes

Depending on the state and the content of its flash memory, the device can enter one of these different boot modes.

- Application code execution

  If the internal flash contains application code with a valid boot header, the application is executed.

- Serial boot

  If no valid boot header is found and the current life cycle phase of the device is less than Infield, an attempt is made to download the application by a serial protocol using the LINFlexD modules or M_CAN modules. The downloaded code is then executed by the boot CPU.

- Static mode

  The device enters power safe mode and waits for an external reset.

52.3.2 Device initialization

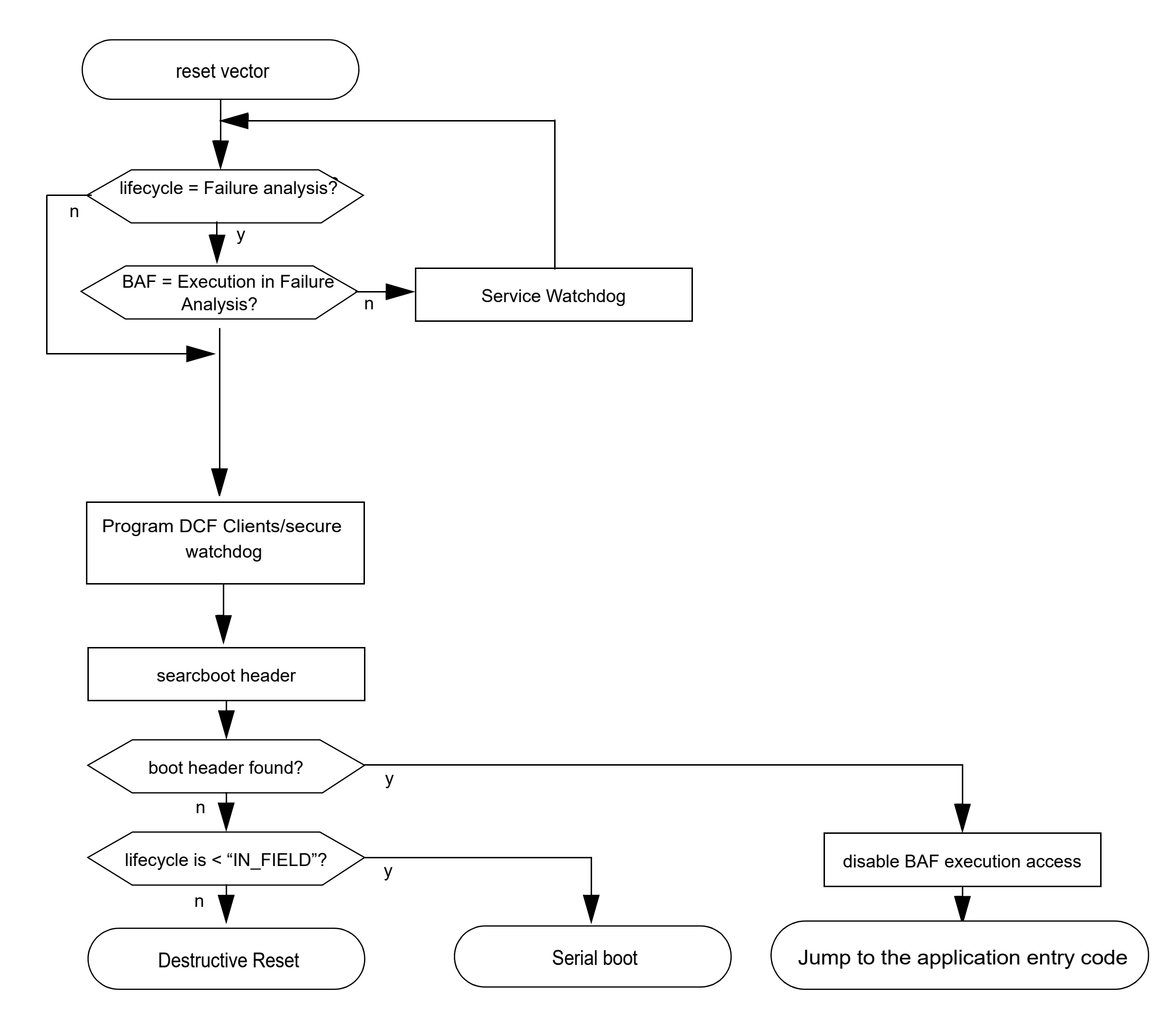
The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact with the Hardware Security Module (HSM) in the following way:

- HSM: If the HSM is enabled and the “wait for HSM” flag is present, then the boot core waits for the HSM to become ready.
- BAF sets up the exception vector table to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors. In the event of an uncorrectable ECC error, the BAF issues a destructive reset from the machine check exception handler if the exception is due to the reading of instructions from the BAF region. If an exception occurs during reading of the Boot header, the BAF ignores the ECC error and jumps to next boot header. If an ECC error occurs during reading of D-MEM, the BAF will still issue a destructive reset. BAF programs the Mode Entry module’s MC_ME_MCTL[TARGET_MODE] to issue a destructive reset.
- BAF uses DMEM memory area of the boot CPU as in *Table 1082: BAF memory organization* for its stack, data and any code that executes from volatile memory. During the BAF start-up procedure, it initializes this section of memory so that there is no ECC error when BAF accesses this memory area.
- The memory area of the BAF is protected from execution for functional safety considerations. This is achieved by disabling BAF execution in PFLASH control register3 (PFLASHC_PFCR3[BAF_DIS]). Finally a mode change (from DRUN to DRUN again) is requested from Mode Entry module to activate the new core configuration. At this point the BAF has finished its operation.

52.3.3 Flow of control

Figure 1055 shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

Figure 1055. Flow of control



52.3.4 Optionally wait for HSM

If the HSM is enabled (SSCM_UOPS[HSE] = 1), the boot core will need to wait for the HSM to become ready (SSCM_UOPS[HSB = 1]). The core then polls the HSM until a ready flag is set. Both SSCM registers can be set by a DCF record. The boot core polls the HSM to HOST FLAGS register (HSM_HSM2HTF) until its least significant bit is set. The boot core also services the watchdog while polling the HSM_HSM2HTF register. This is just in case the HSM is not ready in 16 ms, servicing the watchdog prevents the system going to reset.

52.3.5 Initialization of BAF DCF clients

52.3.5.1 'Soft' DCF clients

DCF clients are 32-bit hardware registers that are written by the SSCM during reset with data that is stored in DCF record format in UTEST region of flash memory. These DCF records are used to configure module registers, and select device operating modes.

The BAF extends the DCF record concept by implementing soft DCF clients. The 64-bit DCF records include:

- 32 bits of data
- 15-bit chip select field (selects the module which implements the DCF clients)
- 15-bit address field (selects a specific DCF client within the selected module)
- Parity bit (not used with UTEST DCF records)
- Stop bit (to indicate the end of DCF record has been reached)

As part of its execution flow, BAF searches through the list of DCF records in UTEST flash to determine if any of the records are to be processed by BAF.

The search starts at one of two addresses: either the start address of DCF records in UTEST (DCF Start Record), or the address specified in UTEST (Soft DCF Record Start Address). If either of the two least significant bits of the 32-bit address stored at the Soft DCF Record Start Address is set, the default start address at the DCF Start Record is used. If the two least significant bits are clear, this 32-bit value is assumed to be the start address of the search.

The default setting of the two least significant bits is 11b (erased value of flash memory is 0xFFFF_FFFF). A user may intentionally program a dummy address with the least two significant bits set in order to disable the search for an alternate address (as the UTEST block is OTP). Valid search start address must start on a 4-byte boundary.

The DCF record search finishes when a DCF record is read with a STOP bit set. The STOP bit may be set because that memory location is not programmed yet (so the list can be added to) or it can be intentionally programmed STOP record to indicate that no more records are added to the list. BAF does not execute the DCF record with STOP bit set. BAF parses the STOP bit before parsing the rest of the records.

BAF uses Chip Select 14 (CS14) for BAF DCF clients. For records with CS14 asserted, the BAF uses the address field (address[14:0]) in DCF records to determine how to process the 32-bits of data. The soft DCF clients programmed by BAF is listed in *Table 1085*.

The DCF clients which are used for initialization of security watchdog are explained in *Section 52.3.5.2: Security watchdog configuration*.

A group of four Soft DCF clients is used to provide a mechanism to write configuration information to address locations. The first DCF client receive 32-bit address information. The next three DCF Clients receive the data that is to be written to the address contained in the first client. The three data clients allow 32-, 16- and 8-bit data to be written to the address in the first client. These four clients can be accessed repeatedly to write whole string of data to memory.

Another soft DCF client as stated in *Table 1085* provides a simple callback feature. The soft DCF record data contains the start address of a subroutine that the BAF calls before it starts searching for the host boot header and exits to customer application code. The callback routine can reside within the BAF or UTEST blocks only. The callback DCF client can be accessed repeatedly in order to execute a callback sequence.

The serial boot DCF client provides a mechanism using which application can inform BAF to perform serial boot or boot from internal flash. Details can be viewed in *Section 52.3.5.6: SER_BOOT_CBACK – BAF serial boot DCF client*.

Table 1085. BAF DCF client list

Address (hex)	Name	Label	Description
0x0000	Security Watchdog Control Register	SEC_SWT_CR	Data to write to SWT Control Register
0x0001	Security watchdog Time out register	SEC_SWT_TO	Security watchdog counter value
0x0002	Security watchdog service address	SEC_IAC8	Instruction Address compare register value - Code present at this address services the watchdog.
0x0003	Security watchdog CPU select	SEC_SWT_CPU	Determines which CPU is to use the security watchdog
0x0004–0x01FF	Reserved		
0x0200	Address	ADDR	The address to which DATAX is written
0x0201	32-bit Data	DATA32	The 32-bit data that is written to ADDR
0x0202	16-bit Data	DATA16	The 16-bit data that is written to ADDR
0x0203	8-bit Data	DATA8	The 8-bit data that is written to ADDR
0x0204–0x020F	Reserved		
0x0210	Callback Address	CALLBACK	The address of a callback function
0x0211	Serial boot Callback Address	SER_BOOT_CBACK	The address of serial boot callback function
0x0212–0x0213	Reserved		Reserved for future use.
0x0214	Reserved		Reserved for future use.
0x0215–0x7FFF	Reserved		Reserved for future use.

Note: BAF boots with a watchdog timeout window set to 16ms. Parsing more and more DCF Records may increase boot time. But it should be taken care that number of DCF records should not be very high as it can cause watchdog window timeout.

52.3.5.2 Security watchdog configuration

BAF initializes the security watchdog before it jumps to the HOST CPU application. BAF implements four DCF clients for security watchdog initialization. BAF parses the DCF records and looks for configuration data to be used to program the security watchdog registers. BAF first collects all information required to program the security watchdog from DCF records and then program the security watchdog registers. BAF configures only SWT2 as security watchdog for the CPU2 core. Configuration of the Security watchdog for other cores has to be performed by the application running on that core as BAF cannot program other core’s IAC8 register. Security watchdog detailed configuration steps are explained in the section below.

52.3.5.2.1 Security Watchdog Control Register (SEC_SWT_CR)

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0000 while parsing, BAF programs the DCF client “Security watchdog control register” with the 32-bit value present in the DATA section of the DCF records. Once written, any subsequent writes to this DCF Client are ignored by BAF.

52.3.5.2.2 Security Watchdog TImeout (SEC_SWT_TO)

The BAF parses the DCF records. While parsing, if the BAF finds a DCF record with CS14 set and address field as 0001h, the BAF programs the security watchdog timeout DCF Clients. This is the security watchdog timeout register. The BAF checks the value obtained from DCF records. If the value yields a timeout greater than 30 seconds, then the BAF changes this value to 30 seconds. Once written, any subsequent writes to this DCF client are ignored.

52.3.5.2.3 Security Watchdog Service Address (SEC_IAC8)

BAF configures the security watchdog in ‘fixed address execution service mode’. In this mode, the watchdog is serviced by executing code at the address loaded into the designated IAC8 (Instruction Address Compare) register. If BAF finds a DCF record with CS14 set and address field as 0x0002 while parsing the DCF records, BAF programs the DCF clients for SEC_IAC8. This DCF client can only be written once. Subsequent writes are ignored.

Note: To program the core IAC8 special purpose register, the ownership of this register should be with software. If hardware has its ownership software will not be able to write to it using mtspr instruction.

52.3.5.2.4 Security Watchdog CPU select (SEC_SWT_CPU)

This DCF client indicates to BAF which CPU will use the security watchdog. BAF parses the DCF record and if it finds a record with CS14 set and address field set as 0003h, BAF programs this DCF client. This means that BAF uses the two least significant bits out of the 32-bit data in the DCF record to find the CPU number which will use the security watchdog. *Figure 1056* and *Table 1086* indicate the data layout for this client in the DCF record. This client is written once during reset. Subsequent writes are ignored.

Writing this client causes the BAF to configure the security watchdog based on data written to the three previous DCF clients. If any of the three previous DCF clients are not written, data is configured erroneously.

During normal BAF operation, SWT2 is configured to perform a conventional task watchdog function (cause reset if not regularly serviced). If the security watchdog is assigned to CPU2 (IOP), first SWT3 is configured to perform the watchdog function task. The timeout value, configuration, and current time is read from SWT2 and copied to SWT3. Then SWT2 is unlocked with current sequence and enabled as security watchdog.

DCF Client Address 0x0003

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				0				0				0			

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				0	0	SC	

Figure 1056. SEC_SWT_CPU bit configuration

Table 1086. SEC_SWT_CPU field description

Field	Description
30:31 SC	These two bits select the CPU which will use the security watchdog. 00 Reserved 01 Reserved 10 CPU2 (IOP) 11 Reserved

Note: BAF programs FCCU for enabling SWT2 and SWT3 as fault source and makes them hardware recoverable. This is done to ensure that fault latched in FCCU get cleared on reset and on next SWT timeout FCCU can again trigger reset. BAF does not clear the faults latched in FCCU as they can be of interest for application.

52.3.5.3 BAF DCF client - ADDRESS

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0x0200 while parsing, BAF programs this DCF client. This client is written with a 32-bit address from the Data section of DCF record. This 32-bit address is used in conjunction with any one of the DATA32, DATA16 or DATA8 DCF Data clients. BAF does not immediately use the ADDRESS information, but it is stored and used later in conjunction with a DATAX DCF client (DATA32, DATA16 or DATA8). Then BAF writes the data to the address in memory specified by this ADDRESS DCF client.

The user can create a series of data writes to memory by programming a series of DCF records consisting of pairs of records. The first DCF record of the pair writes to ADDR to provide the address. The second DCF record of the pair writes to the DATA client (DATAX) to provide the data. Subsequent pairs of DCF records can be programmed to write to more memory locations. The number of pairs of DCF records that can be programmed is only limited by the amount UTEST space set aside for DCF records.

DCF Client Address 0x0200

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADDR															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADDR															

Figure 1057. BAF DCF ADDR client configuration

Table 1087. BAF DCF ADDR client description

Field	Description
0:31 ADDR	32-Bit Address DCF client This ADDRESS value is used in conjunction with a DATAX value. The BAF writes DATAX value to ADDRESS value in the memory space of the MCU.

52.3.5.4 BAF DCF client - DATAX

Caution: While using DATAX DCF records, it should be taken in account that BAF does not take care of ECC scheme and can generate ECC, if write is performed in memory locations which are protected by ECC and are not pre-initialized as required.

52.3.5.4.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0201 while parsing, BAF programs this 32-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant 2 bits to ‘0’ (Only 32-bit aligned writes are supported) and writes this 32-bit data to the address.

DCF Client Address 0x0201

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DATA															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA															

Figure 1058. DATA32 DCF client configuration

Table 1088. DATA32 DCF client configuration

Field	Description
0:31 DATA	32-Bit DCF data

52.3.5.4.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0202 while parsing, BAF programs this 16-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant bits to ‘0’ (Only 16-bit aligned writes are supported) and writes least significant 16-bit data to the address. Data in the 16 most significant bits are ignored.

DCF Client Address 0x0202

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reserved															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA															

Figure 1059. DATA16 DCF Client Configuration

Table 1089. DATA16 DCF client configuration

Field	Description
16:31 DATA	16-Bit DCF data

52.3.5.4.3 DATA8 – 8-bit DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0203 while parsing, BAF programs this 8-bit DCF client. BAF takes the memory address provided by the ADDR client, and writes least significant 8-bit data to the address. Data in the 24 most significant bits are ignored.

DCF Client Address 0x0203

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED								DATA							

Figure 1060. DATA8 DCF Client Configuration

Table 1090. DATA8 DCF client configuration

Field	Description
24:31 DATA	8-Bit DCF data

52.3.5.5 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0210 while parsing, BAF programs this callback DCF client. This client is written with a 32-bit start address of a function which is invoked by BAF. The BAF code checks the start address is within the UTEST or within the BAF flash memory blocks, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory blocks, no function call is executed. The callback is aborted and BAF moves on to the next DCF records. The function that is called is expected to comply with Embedded Application Binary Interface (EABI) for PowerPC found in Power Architecture web site and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function and no return value is expected.

Caution: Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

BAF executes the callback function as the DCF records are parsed. Therefore, several callback functions maybe executed by writing several DCF records that write to the callback DCF clients. The callback functions are executed in the same order as the DCF records.

52.3.5.6 SER_BOOT_CBACK – BAF serial boot DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0211 while parsing, BAF programs this serial boot DCF client. This DCF client is written with a 32 bit start address of function which is invoked by BAF. The BAF code checks the start address is within UTEST or within the BAF flash memory block, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory block, no function call is executed. The DCF client execution is aborted and BAF moves on to the next DCF record. The function that is called is expected to comply with Embedded Application Binary Interface (EABI) for PowerPC found in over Architecture web site and to terminate in orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function.

Warning: No parameters are passed to the function invoked by the BAF.Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

The serial boot callback function returns a char value. The returned value informs BAF if serial boot or normal internal flash boot should be executed (refer to *Table 1091* for details regarding return value).

Table 1091. Serial boot callback function return value definition

Return values	Description
0x00	Boot from internal flash using standard search for boot record (refer to 'Search boot header and boot options' in *Chapter 7: Device configuration*).
0x01	Serial boot using both M_CAN and LINFlexD (in UART mode) (refer to 'Search boot header and boot options' in *Chapter 7: Device configuration*). **Note:** Serial boot is conditioned by life cycle as described in *Figure 1055*

52.3.6 TDM Diary Operation

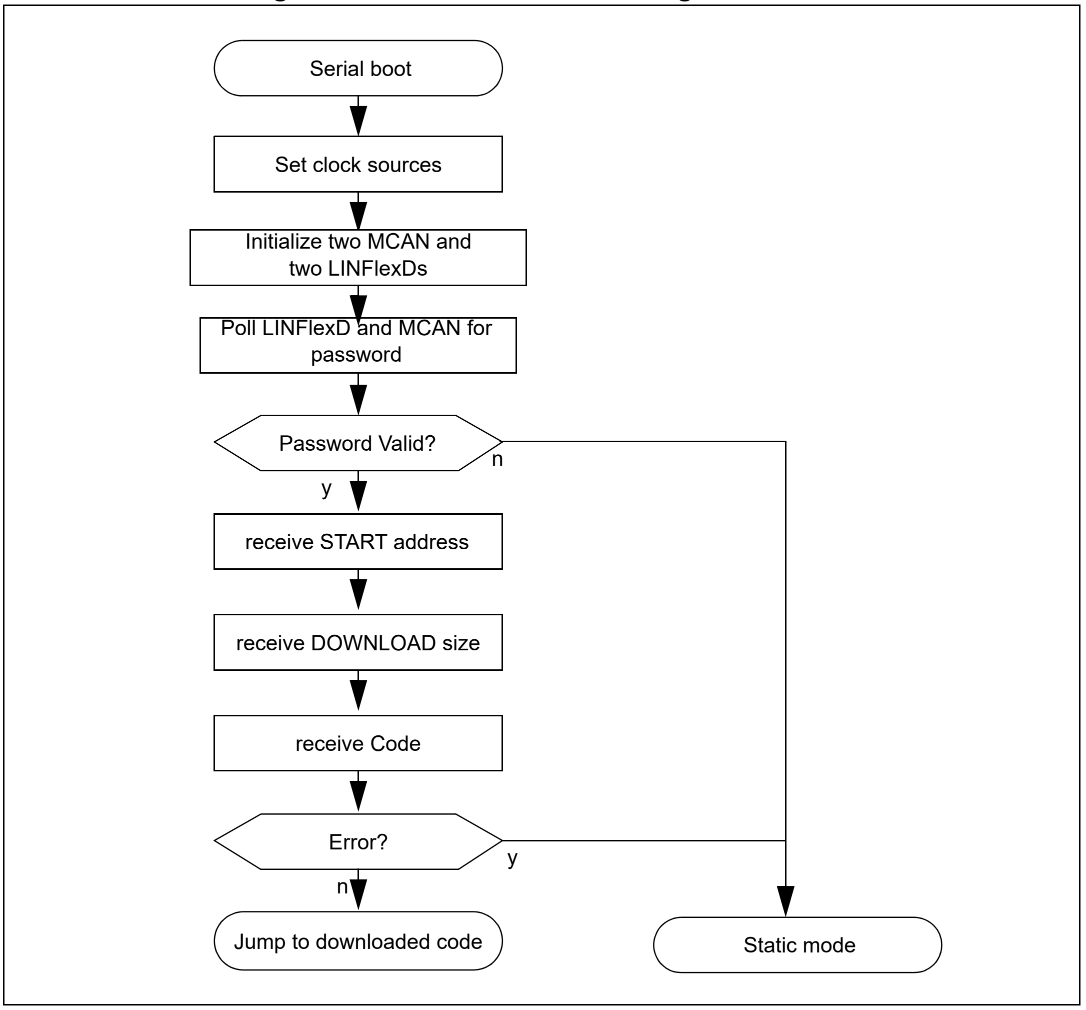
On every reset, BAF copies the first four bytes of last record of every Tamper Detect Region to corresponding Software Tamper Override Key Region Registers in TDM. BAF reads the base address of TDM Diary from DBA register in TDM. For more details on TDM refer to the corresponding section. This is done as a part of TDM Diary Override enhancement.

52.3.7 Optionally perform a serial boot

If no boot header is found in one of the locations mentioned above, then BAF determines the Life Cycle status of the device. If the Life Cycle is lower than In Field, it attempts a serial boot. Otherwise, the boot has failed and the BAF issues a destructive reset.

If the customer selects serial boot using serial boot using serial boot DCF client, then BAF determines the Life Cycle status of the device. If the Life Cycle is lower than Failure Analysis, it attempts a serial boot.

Figure 1061. Flow of control during serial boot



The clock source in the MC_CGM is configured (with IRCOSC for LINFlexD and with XOSC for CAN) and the PADs are set up for two LINFlexDs and two M_CANs signals, then the code is downloaded using LINFlexD or M_CAN in Serial Boot mode. In the absence of errors causing the device to be placed in Static mode, the device state is restored to resemble the out-of-reset state and the downloaded code is executed.

Serial boot allows the download of application to internal SRAM via LINFlexD or M_CAN interface. After downloading the application image to SRAM, the BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At start of serial boot sequence the Rx-channels of all two serial interfaces shall be enabled and the BAF code shall take care for polling all two Rx-channels for detecting the valid serial boot message. After a successful detection of the active Rx-channel the corresponding Tx-channel shall be enabled and the inactive Rx-channel shall be disabled.

At the start of the process both LINFlexDs and M_CANs are configured in passive listening mode, connected to the same pair of pads. As soon as either interface receives a valid message, the receiving interface is configured in full Tx and Rx mode. The non receiving interface is switched back to reset state and does not participate further in serial boot.

If a CAN message of 8 bytes with ID 0x11 is received on the M_CAN controller first, the BAF program

- Disables LINFlexD.
- Configures LINFlexD RXD input mux back to reset state.
- Transitions to CAN serial download protocol routine.

If a CAN message not containing 8 bytes or NOT with ID 0x11 is received by M_CAN controller or a M_CAN controller generates a CAN error, it is probable that the tool is sending a LINFlexD frame. Under such circumstances BAF program

- Ignores the CAN message, if a message arrived.
- Clear any pending CAN error.
- Continue monitoring for valid CAN messages.

If any byte from LINFlexD is received first, the BAF program:

- Checks if the first byte is 0xFE of the password. If not continue polling. The data may be a valid CAN message.
- If it is first byte of the password (0xFE) then disable the M_CAN controller and restore to its reset state.
- Configures the M_CAN pads to their reset state.
- Transition to LINFlexD serial boot (UART mode).

52.3.7.1 Serial boot mode protocol

From a high-level perspective, the download protocol over M_CAN and LIN interfaces follows the steps below:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes, NO_ECHO bit and VLE bit to MCU.

   NO_ECHO bit signifies if actual download data to be echoed or not. If NO_ECHO bit is set, then data will not be echoed otherwise if NO_ECHO bit is clear, then data will be echoed.

Note: Since this device supports only VLE code and no Book E code, the VLE bit is used only for backward compatibility.

3. Host transmits a sequence of data bytes that are to be executed.
4. The boot CPU executes code from start address provided in step 2.

Each step must be completed before the next step starts.

The exact protocol between LINFlexD and M_CAN are slightly different.

52.3.7.2 LINFlexD download protocol

The communication is performed in half-duplex mode, any transmission from the host is followed by the MCU transmission.

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received 8-byte of password, 4-byte of load address and 4-byte of VLE-bit, NO_ECHO bit and 30-bit of download size to the host. The MCU echoes actual download data depending upon NO_ECHO bit.
3. The host verifies whether the echo is correct.
   – If the data is correct, the host can send the next byte of data.
   – If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

The tool sends data in the following order:

1. 8 bytes that represent the password.
2. 4 bytes that represent the start address where the code is to be downloaded.
3. 4 bytes that represent the number of bytes to be downloaded, NO_ECHO bit and the VLE bit.
4. *x* bytes of code that is downloaded to SRAM and executed (*x* must match the value specified in step 3.).

52.3.7.3 M_CAN download protocol

The M_CAN download protocol uses very similar structure to the LINFlexD protocol, but it makes use of the message ID's provided by M_CAN.

The tool sends data packed into standard CAN message in the following manner:

1. A message with 0x11 as the ID and an 8-byte length to send the password. The MCU transmits the same message, but with an ID of 0x1.
2. A message with 0x12 as the ID and 8-byte length to send the start address, size of the code and VLE mode bit.The MCU transmits back the same data but with an ID of 0x2.
3. A message with 0x13 as the ID is used to send data that is to be downloaded to SRAM. The MCU transmit the same data with an ID of 0x3.

52.3.7.4 Download 64-bit password and password check

The first 64 bits received represent the password. The MCU only supports public passwords. The received password is always compared with the public password 0xFEED_FACE_CAFE_BEEF.

- If the correct password is received, BAF continues its task.
- If an incorrect password is supplied, BAF puts the device into static mode (low power safe mode).
- If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU.

52.3.7.5 Download start address, VLE bit, NO_ECHO bit and code size

After the password, the next 8 bytes received by the MCU contains a 32-bit start address, the VLE mode bit, NO_ECHO bit and 30-bit code length as shown in *Figure 1062* and *Figure 1063*.

The VLE bit (Variable Length Encoding) is used to indicate for which instruction set the downloaded code is compiled. The UART serial boot protocol supports download of VLE Code (VLE = 1) and Book E Code (VLE = 0).

The MCU only supports VLE code. If the BAF detects that the host is sending Book E code, the BAF puts the device into static mode.

The NO_ECHO bit is used to indicate if download data should be echoed by the MCU. If the NO_ECHO bit is set (for example, NO_ECHO = 1), then only 8 bytes of password, 4 bytes of download address, and 4 bytes of download size are echoed, but download data is not echoed. If the NO_ECHO bit is clear, then every byte sent to the MCU must be echoed.

The start address defines where the received data is stored and where the MCU branches after the download is finished. The three LSB bits of the start address are ignored by BAF, such that the load address is 64-bit double-word aligned.

The length determines how many data bytes has to be loaded.

Bytes 0:3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
START_ADDRESS															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
START_ADDRESS															

Figure 1062. Start address

Bytes 4:7

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VLE	NO_ECHO	CODE_LENGTH													

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CODE_LENGTH															

Figure 1063. VLE bit, NO_ECHO bit and code size

52.3.7.6 Download data

Each byte of data received is downloaded into device SRAM starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until a number of bytes of data received matches the code length specified in the previous protocol step.

Since the SRAM is protected by 64-bit wide Error Correction Code (ECC), BAF always writes to SRAM grouped in 64-bit double-words. If the last byte does not fall into a 64-Bit bounder, the BAF fills it with zero’s and renders the data aligned on a 64-bit bounder. After the last data is written to SRAM, the BAF writes a dummy double-word (0x0000_0000_0000_0000) to the next address location. This is written to avoid possible ECC error during core prefetch.

52.3.7.7 Execute code

The BAF program waits for the last echo message transmission to be completed. Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which was received in step 2 of the protocol. At this point the BAF has finished its tasks and the MCU is controlled by new code executing from SRAM.

52.3.8 Serial boot configuration

52.3.8.1 LINFlexD configuration

Boot using UART protocol is implemented by LINFlexD module.

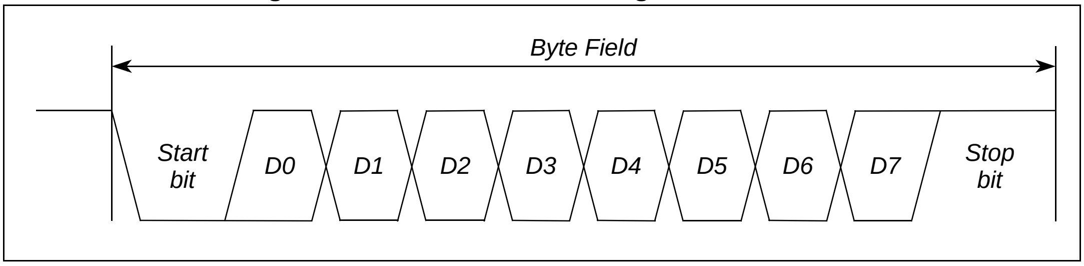
Note: Refer to the pin configuration table in Chapter 7: Device configuration for pins used by the LINFlexD module.

When Serial Boot mode is started the LINFlexD_Rx is configured as an input and multiplexed to the LINFlexD module. The LINFlexD_Tx remains in a passive input state until the first byte of protocol is received. It is then configured as an output and multiplexed to LINFlexD module. The application may not be using the Serial Boot Mode function, and may be using the pin used by LINFlexD_Tx as an input, which may have some external hardware driving the pin. Inadvertently driving this pin as an output would cause contention and possible damage.

The LINFlexD module operates in half-duplex mode. The interface is either transmitting a byte or receiving a byte, but never both at the same time. It is expected that LINFlexD pins are connected to a physical layer bus driver/receiver (PHY). Many PHYs enforce half-duplex mode by using a single data channel for both transmitting and receiving data . The side effect of this is that all data transmitted on Tx pin is also received on Rx pin. To prevent the LINFlexD module receiving its own transmission, the Rx pin is blocked during Tx pin transmission.

The LINFlexD controller is configured to operate with an 8-bit data frame without parity bit and 1 stop bit.

Figure 1064. LINFlexD bit timing in UART mode

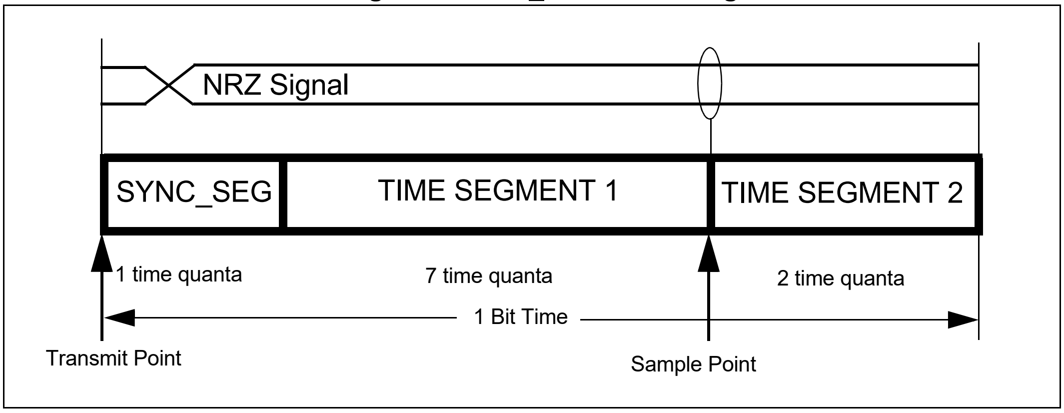


52.3.8.2 M_CAN configuration

The M_CAN controller is configured to operate at a baud rate equal to crystal frequency divided by 40, using the standard 11-bit identifier format detailed in CAN 2.0A specification (refer to *Table 1092* for baud rates).

The Bit timing is configured as shown in *Figure 1065*

Figure 1065. M_CAN Bit Timing



52.3.8.3 Baud rates

The LINFlexD baud rate is set at 250kbaud. *Table 1092* below shows some possible baud rates.

The M_CAN baud rate is set by the frequency of the external crystal/40.

Table 1092. M_CAN baud rates

Frequency	M_CAN baud rate
8 MHz	200 Kbaud
12 MHz	300 Kbaud
16 MHz	400 Kbaud
20 MHz	500 Kbaud
40 MHz	1000 Kbaud

52.3.8.4 LINFlexD and M_CAN pin configuration

Refer to *Chapter 7: Device configuration* for pin details.

52.3.8.5 Protocol summary

Table 1093 summarizes the LINFlexD protocol and BAF action during this boot mode.

Table 1093. LINFlexD serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	64-bit password (MSB first)	64-bit password (MSB first)	Password checked for validity and compared with stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.

Table 1093. LINFlexD serial boot mode download protocol (continued)

Protocol steps	Host message sent	BAF response message	Action
3	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	If NO_ECHO bit is set, data is not echoed, otherwise data is echoed.	8 x 8 bits of data are packed into 64-bit double-words. These double-words are stored in SRAM starting from the Load address. The Load address increments until the number of data received and stored matches the size as specified in the previous steps.
5	None	None	Branch to downloaded code

Table 1094 shows the M_CAN serial boot protocol.

Table 1094. M_CAN serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	CAN ID 0x011 + 64-bit password	CAN ID 0x001 + 64-bit password	Password checked against fixed password (0xFEED_FACE_CAFE_BEEF). Watchdog timer is refreshed if the password check is successful.
2	CAN ID 0x012 + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 of download data size in bytes)	CAN ID 0x002 + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 of download data size in bytes)	Load address and size of download are stored for future use.
3	CAN ID 0x013 + 8 to 64-bits of raw binary data	VLE bit + 31 bit number of bytes (MSB first)	Each byte of data received is store in MCU memory, starting at the address specified in the previous step and incrementing until the amount of data received and stored, matched the size as specified in the previous step.
4	None	None	The BAF returns IO pins and CAN module to their reset state, then branches to the first address the data was stored to (As specified in step 2)
5	None	None	Branch to downloaded code

52.4 Resources accessed by BAF code execution

The BAF code execution uses several IPs to achieve the main functions.

- Magic carpet (clocks)

Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before execution of the application code (in flash memory or received via serial download).

- LINFlexD and M_CAN

The LINFlexD module is only used if normal boot header search fails and a serial download boot is attempted, or from the SER_BOOT_CBACK callback function. All settings are restored before the application code in system RAM is executed.

- PFLASH

PFLASH module is used to disable BAF execution. Remapping of the external overlay memory is configured automatically under certain conditions. The two PFLASH are used.

- Interrupts

No interrupts/exceptions are used or enabled, except the Machine Check exception, to handle ECC errors.

- Security watchdog

When SWT2 is chosen as security watchdog, SWT3 is used for task monitoring purpose.

- FCCU

FCCU is used to configure Fault Reaction of SWT2 and SWT3.

- TDM

TDM for implementing TDM Diary enhancements.

- SSCM

Used to read some register bits (mirrors of DCF)

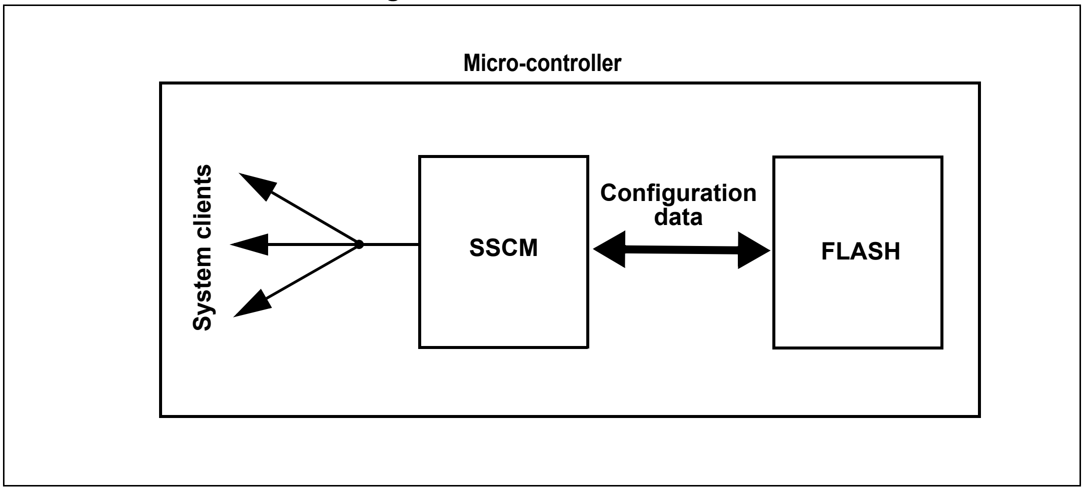
53 System Status and Configuration Module (SSCM)

53.1 Introduction

53.1.1 Overview

The System Status and Configuration Module (SSCM) is aimed at reading the system configuration from the Flash and distributing to various clients inside the micro-controller, pictured in *Figure 1066*.

Figure 1066. SSCM overview



53.1.2 Features

The SSCM includes these distinct features:

- System Configuration and Status
  - Device Mode and System Status
  - Determine primary boot vector
  - Determine HSM boot vector
  - Decodes the Device Life Cycle

53.1.3 Modes of operation

The SSCM operates identically in all system modes.

53.2 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

Table 1095 shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

Note: All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit *boundaries. As an example, the STATUS register is accessible by a 16-bit READ/WRITE to address Base + 0x0002, but performing a 16-bit access to Base + 0x0003 is illegal.*

Table 1095. SSCM memory map

Address offset	Register	Access	Reset value	Section
0x0000	SSCM System Status (STATUS)	R/W	0xyyy0(1)	Section 53.2.1.1
0x0002	SSCM System Memory and ID Register (MEMCONFIG)	R	0xyyyy(1)	Section 53.2.1.2
0x0004–0x0005	Reserved			
0x0006	SSCM Error Configuration Register (ERROR)	R/W	0x0000	Section 53.2.1.3
0x0008–0x0009	Reserved(2)			
0x000A–0x001C	Reserved			
0x0020	SSCM HSM and User Option Status Register (UOPS)	R	0xyyyy_yyyy(1)	Section 53.2.1.4
0x0024–0x0027	Reserved			
0x0028	Processor Start Address Register (PSA)	R	0xyyyy_yyyy(1)	Section 53.2.1.5
0x002C–0x002F	Reserved			
0x0030	SSCM HSM Start Address Register (HSA)	R	0xyyyy_yyyy(1)	Section 53.2.1.6
0x0034	Life Cycle Status Register (LCSTAT)	R	0x0000_0y0y(1)	Section 53.2.1.7
0x0038–0x004F	Reserved			

1. Refer to register description for reset value details.
2. No register abort on this location.

53.2.1 Register descriptions

The following registers are available in the SSCM. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits should be masked out during any read/write operations to avoid conflict with future revisions.

53.2.1.1 System Status Register (STATUS)

The System Status register reflects the current state of the system.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CER	0	NXEN1	NXEN	1	0	0	BMODE			VLE	0	0	0	0
W		w1c														
Reset	0	0	0	-(2)	-(2)	1	0	0	-(2)			-(1)	0	0	0	0

1. Reset value depends on the associated option bits in flash memory.
2. Reset value depends on the device status after leaving reset.

Figure 1067. System Status Register (STATUS)

Table 1096. STATUS field descriptions

Field	Description
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error while loading DCF clients (with PARITY Enabled). This error may have been detected during the reset sequence. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct If a non-permanent (transient) error is detected, this bit can be cleared by writing a 1. If the error is permanent, the bit will reappear immediately.
3 NXEN1	Processor 1 Nexus enable status 0 Processor 1 Nexus disabled. 1 Processor 1 Nexus enabled
4 NXEN	Processor 0 Nexus enable status. 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
8:10 BMODE	Device Boot Mode 000 Reserved 001 Reserved 010 Reserved 011 User Code Boot (BAF not executed) 100 Reserved 101 BAF Code Boot (BAF executed) 110 Reserved 111 Reserved This field is only updated during reset. If the device goes into standby mode and wakes up from it again, the bits retain their original value.
11 VLE	Variable Length Instruction Mode When booting in User Code Boot mode, this field indicates that the code stored in the flash memory is using the VLE instruction set. The value of this field is determined by the RCHW field of the flash memory boot sector. 0 User Code Boot Location contains standard PPC code 1 User Code Boot Location contains VLE code

53.2.1.2 System Memory and ID Register (MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG ID.

Offset: 0x0002 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	JPIN										1	MREV				0
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	1	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0

1. Reset value reflects the JTAG ID of the device.

Figure 1068. System memory and ID register (MEMCONFIG)

Table 1097. MEMCONFIG field descriptions

Field	Description
0:9 JPIN	JTAG Part ID Number
11:14 MREV	Minor Mask Revision

53.2.1.3 Error Configuration Register (ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Offset: 0x0006 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PAE	RAE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1069. Error Configuration Register (ERROR)

Table 1098. ERROR field descriptions

Field	Description
14 PAE	Peripheral Bus Abort Enable This bit enables bus aborts on: – any access to a peripheral slot that is not used on the device. – illegal accesses to off-platform peripherals. On platform peripherals cannot be controlled by SSCM peripheral abort. 0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception
15 RAE	Register Bus Abort Enable This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. 0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception **Note:** Transfers to peripheral bus resources may be aborted even before they reach the peripheral bus (in example, at the PBRIDGE level). In this case, the PAE and RAE register bits have no effect on the abort.

53.2.1.4 HSM and User Option Status Register (UOPS)

Offset: 0x0020 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BAF_FA	FL_STDBY	FL_RUN	0	0	0	0	0	0	0	0	0	HSB			HSE
W																
Reset	0/1[(1)]	0	0	0	0	0	0	0	0	0	0	0	0/1[(1)]	0/1[(1)]	0/1[(1)]	0/1[(1)]

1. Values are determined by DCF record and life cycle - default is 0.

Figure 1070. HSM and User Option Status Register (UOPS)

Table 1099. UOPS field descriptions

Field	Description
16 BAF_FA	BAF execution mode when bypass mode DCF is zero and lifecycle is FA 0 BAF enters an infinite loop 1 BAF executes as normal **Note:** This bit is mirrored from bit 31 of the UTEST_Miscellaneous_2 DCF client.
17 FL_STDBY	Code flash on after standby mode, despite MC_ME programming 0 Code Flash can be switched off when exiting from STANDBY mode, if programmed in MC_ME.MC_DRUN.CFLAON 1 Code Flash cannot be switched off when exiting from STANDBY mode, even if programmed in MC_ME.MC_DRUN.CFLAON **Note:** This bit is mirrored from bit 30 of UTEST_Miscellaneous_2 DCF client. When this bit is set, writing on MC_ME.MC_DRUN.CFLAON has no effect.
18 FL_RUN	Code flash on during RUNx mode, despite MC_ME programming 0 Code Flash can be switched off in RUNx mode, if programmed in MC_ME.MC_DRUx.CFLAON 1 Code Flash can be switched off in RUNx mode, even if programmed in MC_ME.MC_RUNx.CFLAON **Note:** This bit is mirrored from bit 29 of UTEST_Miscellaneous_2 DCF client. When this bit is set, writing on MC_ME.MC_RUNx.CFLAON has no effect.
28:30 HSB	HSM Boot Configuration 001 When the device is in BAF Code Boot mode, BAF waits for HSM ready flag before allowing application code to start. In User Code Boot mode the SSCM waits for HSM ready flag before allowing application code to start. Others: When the device is in BAF Code Boot mode, BAF will not wait for HSM ready flag before allowing application code to start. In User Code Boot mode the SSCM will not wait for HSM ready flag before allowing application code to start. This field can be set via DCF record (refer to the SPC584Bx Microcontroller Security Reference Manual for the HSM details).
31 HSE	HSM Enabled This bit is set to 1 if HSM is enabled via DCF record.

53.2.1.5 Processor Start Address Register (PSA)

Offset 0x0028 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SADR[(1)]															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADR[(1)]															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

1. Reset value is the start address.

Figure 1071. Processor Start Address Register (PSA)

Table 1100. PSA field descriptions

Field	Description
0:31 SADR	Start Address The boot processor will start executing application code from this address. In User Code Boot mode this indicates the location determined by the RCHW search. In BAF Code Boot mode this will show the start address of the BAF code. In the case of a serial download, this register will point to the BAM start location.

53.2.1.6 HSM Start Address Register (HSA)

Offset 0x0030 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HADR															
W																
Reset	0/1[(1)]	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HADR															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Figure 1072. HSM Start Address Register (HSA)

Table 1101. HSA field descriptions

Field	Description
0:31 HADR	HSM Start Address HSM will start executing application code from this address. The value of this register is determined by the HSM boot header search.

53.2.1.7 Life Cycle Status Register (LCSTAT)

Offset 0x0034 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	LC		
W																
Reset	0	0	0	0	0	0	0/1(1)	0/1(1)	0	0	0	0	0	0/1(2)	0/1(2)	0/1(2)

1. Reset value reflects the Test Life Cycle of the device.
2. Reset value is the Life Cycle of the device.

Figure 1073. Life Cycle Status Register (LCSTAT)

Table 1102. LCSTAT field descriptions

Field	Description
29:31 LC	Life Cycle As determined by the SSCM by reading the LC slots. The encoding is: 000 FAIL_ANALYSIS 001 Reserved 010 PROD_OEM 011 CUST_DELIV 100 Reserved 101 Reserved 110 ST_PROD 111 IN_FIELD

53.3 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

53.3.1 DCF mechanism

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device will process these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

Each record is 64 bits in length. *Figure 1074* shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records should be processed.

Figure 1074. DCF start record

0:31
0x05AA 55AF

0:31
0x0000 0000

The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - refer to *Figure 1075*. In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Figure 1075. Format of a DCF data record

0:31
WDATA[31:0]

0:14	15:29	30	31
CS[14:0]	ADDR[16:2]	PRTY	STOP

The stop record indicates that processing should stop. The general format of the stop record is shown in *Figure 1076*. Only the STOP bit needs to be 1 in order to form a stop record – All other bits are ignored. An unprogrammed location in UTEST flash will be interpreted as a stop record.

Figure 1076. DCF stop record

0:31
Reserved

0:30	31
Reserved	1

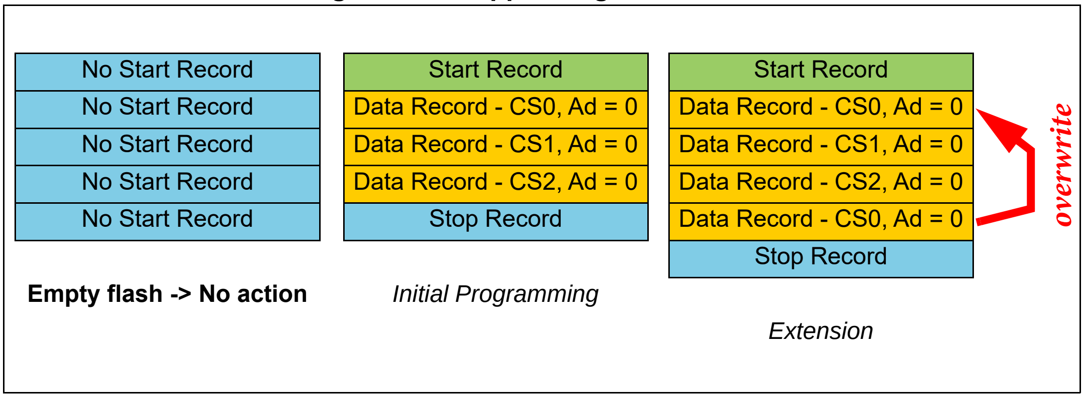
If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 1103*.

Table 1103. Series of DCF records in UTEST

ADDR offset	DATA			
0x00	0x05AA_55AF			
0x04	0x0000_0000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8n - 1 + 0x0	Reserved			
8n - 1 + 0x4	Reserved			1
8n + 0x0				
8n + 0x4				

There must never be an unprogrammed record in the data structure, as that would be interpreted as a stop record, so subsequent records would be ignored. This allows programming the records in several sessions, each time appending new records at the end of the list, as shown in *Figure 1077*.

Figure 1077. Appending DCF records



In this case, a record may overwrite a client's value which was already set by a previous record. However whether such an operation is allowed depends on the rules given for the client - some clients are write-once, only allow setting bits but not clearing, and so on.

The parity bit (PRTY) is required for some DCF records. The parity scheme used is even parity, so the number of 1s in the WDATA and PRTY fields needs to be even. (For example, if the WDATA field has the value 0x0000_0001 then the PRTY field needs to be set to '1', so that the total number of 1s is even).

53.3.2 ECC error monitoring

During flash memory scanning (such as Life Cycle or DCF records), the SSCM will monitor the flash memory status signals for ECC and other unexpected access errors. If such an error occurs the SSCM will request a destructive reset.

53.3.3 Boot mode functionality

The device supports the following boot modes for the main boot core:

- User Code Boot mode (SSCM): the device will boot from the first bootable section of the flash memory main array (BOOT_ID = 0x5A).
- BAF Code Boot: the device will boot from UTest NVM Flash Block, (interface with the HSM if available) and perform device setup (BOOT_ID = 0xA5).
- Standby-Boot : after waking up from Standby Mode, the processor will execute from the Standby RAM. The flash memory will not be accessible in this mode.

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location for SSCM and BAF, that is 0x5A and 0xA5 respectively) then the device will enter serial boot mode.

If the HSM is enabled via DCF record, the HSM will boot from the location found via the HSM boot header search. Refer to *Section 53.3.6: HSM boot header search* for details.

53.3.4 BAF configuration

In BAF Code Boot mode the boot core will start to execute the BAF code at the location provided in the “Embedded Flash Memory” chapter (TestFlash block memory map). The BMODE field in the SSCM status register will be set to the value 0b101. Details of the BAF operation are described in the BAF chapter.

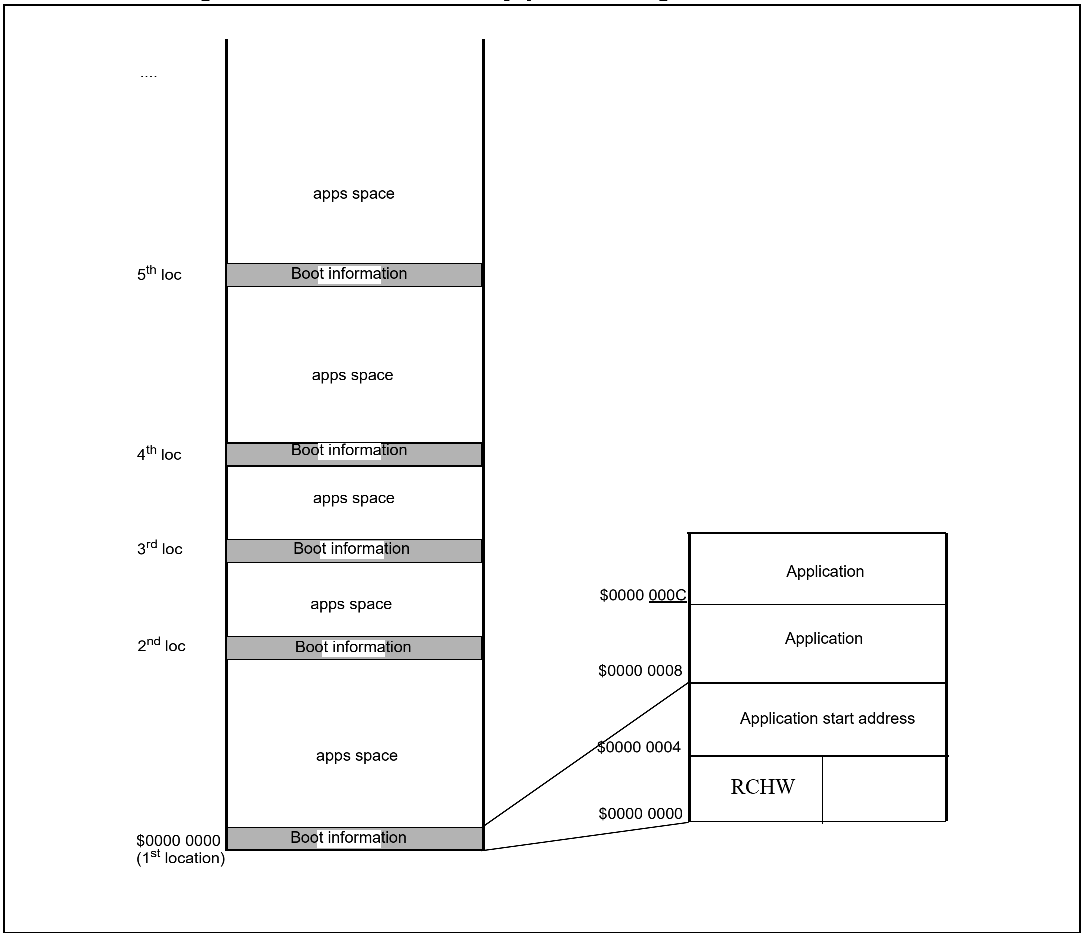
53.3.5 User code boot sector search

53.3.5.1 Potential boot sectors

As shown in *Figure 1078* in User Code Boot mode, several locations are searched for a valid boot ID. The lowest sector which starts with a valid boot ID will be used to boot the device.

The flash memory locations that are searched for boot information are shown in the “Reset and Boot” chapter (section “HW boot record search (SSCM search)”).

Figure 1078. Flash memory partitioning and RCHW search



53.3.5.2 Reset Configuration Half-Word

For each boot sector in the flash memory, SSCM checks the content of the Reset Configuration Half-Word (RCHW) at offset 0x00. If the RCHW field BOOT_ID holds the value 0x5A then the sector is considered bootable. In addition, there is a flag which indicates that the code is VLE code. (If a device only supports VLE code for future compatibility the flag must always be set to '1'.) All other bits are reserved.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	VLE	BOOT_ID							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved

Figure 1079. Reset Configuration Half-Word

Once the device detects that it needs to boot from flash memory, and finds a valid BOOT_ID, it will boot from the application start address at offset 0x04 within the boot sector.

53.3.5.3 Standby Boot mode

In Standby Boot mode, the MCU can be configured to boot from internal flash memory (User Code Boot mode) or from Standby RAM. If the MCU boots from Standby RAM the reset sequence can be abridged - in this case the flash memory will not be available (refer to RGM block guide for mode selection).

The device security status will be stored in the SSCM Standby area, so debugging will be possible if the device is unsecured.

To enter Standby Boot mode (SBM) first the RGM needs to be configured appropriately, then low power standby mode must be entered and subsequently a wakeup interrupt must be received.

The intent of SBM is to allow quick wakeup from low power standby mode. Upon wakeup application code could perform a quick check whether the device is needed, perform a short task, then return to standby again and conserve power.

To further aid this, the device allows to boot without activating the flash memory - which significantly reduces the length of the wakeup sequence (refer to the MC_RGM chapter).

In Standby Boot mode the device will boot from Standby RAM offset 0x00. The application needs to load code to this location before going into low power standby mode. In some cases the entire code will be located here, but it is also possible to merely place a jump command at offset 0x00 and continue execution from another location (for example, the flash memory if it is activated or from the external bus).

If the device boots without enabling the flash memory, it is not possible to determine the security status again. Hence the security status is stored in the SSCM standby area, and the device will therefore have the same access restrictions as were in place before entering low power standby mode. For example, if the device was secure before entering low power standby mode, then debug will not be possible in Standby Boot mode.

Sometimes it is desirable to activate the flash memory even though it was initially bypassed in Standby Boot mode. To do this the application needs to select a run mode which enables the flash memory - the device will transition into that run mode once the flash memory reset sequence is complete.

53.3.6 HSM boot header search

The SSCM searches the code flash memory of the HSM module for a valid boot header (refer to the SSCM section of the “Device Configuration” chapter for address(es)) and provides the boot vector to the CPU of the HSM module.

The HSM Boot Header has the format as described in *Figure 1080*.

Figure 1080. Boot header structure

0x00	Boot Header ID	
0x08	Reserved	Start Address
10x0	Reserved for Configuration Bits	

A valid Boot Header ID has the value 0xFFFF_0000_FFFF_0000.

The first valid Boot Header found is the one which is passed to the HSM CPU. In order to switch from one Boot Header to a Boot Header at a higher address, the Boot Header at the lower address must be invalidated. This can be achieved by over-programming the Boot Header ID location with 0x0000_0000_0000_0000. Since this value results in the same ECC encoding it will still be possible to read the location without causing an ECC error.

53.3.7 Life Cycle

The SSCM will determine the Life Cycle of the device by reading the Life Cycle slots from UTEST flash memory area. The read operation is done during the reset phase with normal timings and it is protected by both operating monitors and ECC check. In addition a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- ST Production
- Customer Delivery
- OEM Production
- In Field
- FA

The LC is written into 5 slots, 256 bits each, at fixed locations in UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- The valid field
- The invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 status:

Table 1104. Life Cycle slots

Use case	Offset	LC slot	Valid field (128 bits)		Invalid field (128 bits)		LC slot status
1	0x00-0x10	1	Erased	Erased	Erased	Erased	Erased
2	0x00-0x10	1	Marked	Erased	Marked	Erased	Inactive
3	0x00-0x10	1	Marked	Erased	Erased	Erased	Active
4	0x00-0x10	1	Any other value				Illegal

“Marked” in this case means that the value has been programmed with the bit pattern 0x55AA_50AF_55AA_50AF, “Erased” is detected by the bit pattern 0xFFFF_FFFF_FFFF_FFFF.

Table 1105 shows how the slots are arranged in memory.

Table 1105. Life Cycle Slots in Memory

Offset	LC Slot word
0x00 - 0x07	Valid[63:0]
0x08 - 0x0F	Don't care

The Life Cycle is determined as shown in *Table 1106*. The priority of the entries is from top to bottom, so the first row which applies determines the resulting Life Cycle.

Table 1106. Resulting Life Cycle

LC Slot 0 ST_PROD(1)	LC Slot 1 CUST_DEL(1)	LC Slot 2 OEM_PROD(1)	LC Slot 3 IN_FIELD(1)	LC Slot 4 FA(1)	Resulting LifeCycle
Active	Erased	Erased	Erased	Erased	ST Production
Inactive	**Active**	Erased	Erased	Erased	Customer Delivery
Inactive	*Inactive*	**Active**	Erased	Erased	OEM Production
Inactive	*Inactive*	*Inactive*	**Active**	Erased	In Field
Inactive	*Inactive*	*Inactive*	*Inactive*	**Active**	FA
Any other combination				Erased	In Field
Any other combination					System Reset

1. Erased = all slot bits at ‘1’

53.4 Initialization and application information

53.4.1 Reset

The reset state of each individual bit is shown within the Register Description section (refer to *Section 53.2.1: Register descriptions*).

53.5 Additional safety measures

53.5.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM but not other modules) by gating the state machines with the expected status of the MC_RGM. In case of a spurious reset the SSCM will not interfere with the flash memory bus or overwrite configuration registers. Status internal to the SSCM will be lost, however.

54 Power management controller digital interface (PMC_Dig)

54.1 Introduction

The Control PMU manages the PMU at reset, standby entry, standby exit and during run. It checks the LVD or HVD accuracy.

The PMC analog block interfaces the digital part of the device by the PMC_Dig block.

This chapter describes the digital block of the PMU.

The PMC_Dig contains all of the registers and digital logic to generate all of the enable signals, two stage trim control bits, Power on Reset (POR) generation, standby entry/exit sequence and test mode logic for the analog block. The PMU digital logic also contains the controls for PMU analog outputs to be sensed with the ADC module.

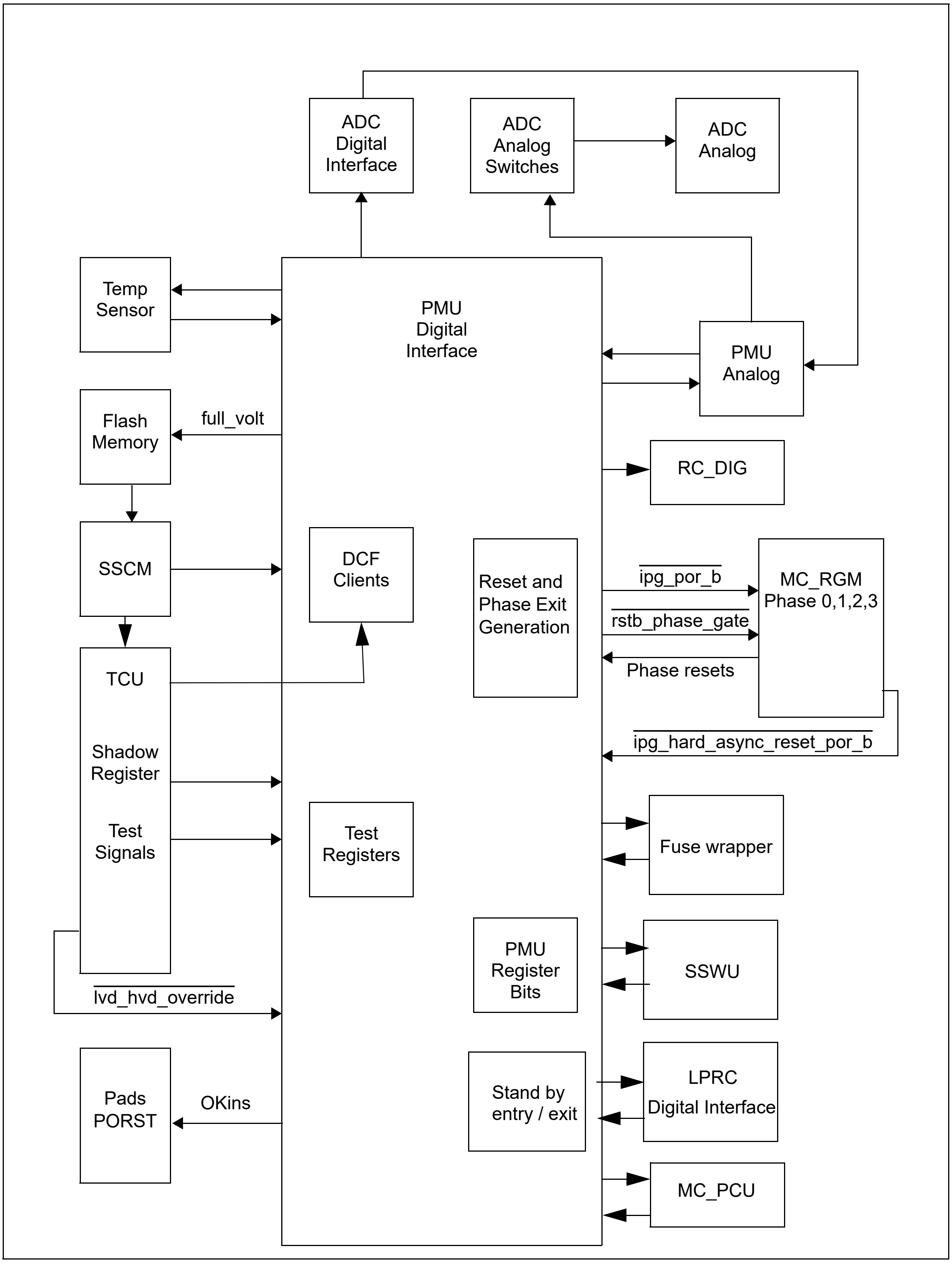
In addition, the PMU digital logic contains the enable signals, trim bits and other registers for the temperature sensor for SOC and a dedicated temperature sensor for RC regulator.

There are also several custom interfaces to various other blocks: The Reset Generation Module (MC_RGM) block receives the LVD values during POR to use for phase transition conditions and reset generation of FUSE_WRAPPER.

There are two custom interfaces, one to Flash memory via the SSCM (and DCF client blocks) that is used to load the trim values during initial POR. The other to fuse block via fuse_wrapper that is responsible for first stage trimming, that is to say, bandgap trimming of both regulator and monitor.

The PMU digital block generates a transfer error event to the system if the selected address is greater than the highest address in the PMU digital memory map. A transfer error is not generated if an unused address within the PMU digital memory space is accessed. *Figure 1081* shows the top level block diagram of the digital PMU and its interfaces to the device modules.

Figure 1081. Digital PMU block diagram



54.2 Main features

- PMU management at a main reset
- Resets signal generation

54.2.1 Standard features

- IPS bus interface

54.3 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported:

- word (32 bits) data write/read operations to any registers
- low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data write/read operations to any registers
- any other operation (free byte enables or other operations) must be avoided.

The PMC_Dig block generates a transfer error in the following cases:

- any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral.
- any write/read operation different from byte/hword/word (free byte enables or other operations) on each register.

The registers of the PMC_Dig block are accessible (read/write) in each access mode: user, supervisor or test.

About the Reset used among the register:

- The POR reset is used on EPR_LVx, EPR_HVx, REE_LVx, REE_HVx, RES_LVx and RES_HVx registers
- The destructive reset generated by voltage detectors is used on REE_TD, RES_TD, FEE_LVx, FEE_HVx, BIST_x and FEE_TD registers
- The functional reset is used on IE_HVx, IE_LVx, IE_G, CTL_TD registers

54.4 Memory map/register definition

Table 1107 shows the PMU memory map. The PMU includes many registers for configuring and monitoring LVD monitors, and some other control registers.

Table 1107. Power Management Controller Memory Map

Offset	Register name	Location
0x0000	Event Pending Register LV0 (EPR_LV0)	*Section 54.5.1*
0x0004	Reset Event Enable LV0 (REE_LV0)	*Section 54.5.2*
0x0008	Reset Event Select LV0 (RES_LV0)	*Section 54.5.3*
0x000C	Interrupt Enable LV0 (IE_LV0)	*Section 54.5.4*

Table 1107. Power Management Controller Memory Map (continued)

Offset	Register name	Location
0x0010	FCCU Event Enable LV0 (FEE_LV0)	Section 54.5.5
0x0014–0x001F	Reserved	
0x0020	Event Pending Register LV1 (EPR_LV1)	Section 54.5.6
0x0024	Reset Event Enable LV1 (REE_LV1)	Section 54.5.7
0x0028	Reset Event Select LV1 (RES_LV1)	Section 54.5.8
0x002C	Interrupt Enable LV1 (IE_LV1)	Section 54.5.9
0x0030	FCCU Event Enable LV1 (FEE_LV1)	Section 54.5.10
0x0034–0x003F	Reserved	
0x0040	Event Pending Register HV0 (EPR_HV0)	Section 54.5.11
0x0044	Reset Event Enable HV0 (REE_HV0)	Section 54.5.12
0x0048	Reset Event Selection HV0 (RES_HV0)	Section 54.5.13
0x004C	Interrupt Enable HV0 (IE_HV0)	Section 54.5.14
0x0050	FCCU Event Enable HV0 (FEE_HV0)	Section 54.5.15
0x0054–0x005F	Reserved	
0x0060	Event Pending Register HV1 (EPR_HV1)	Section 54.5.16
0x0064	Reset Event Enable HV1 (REE_HV1)	Section 54.5.17
0x0068	Reset Event Selection HV1 (RES_HV1)	Section 54.5.18
0x006C	Interrupt Enable HV1 (IE_HV1)	Section 54.5.19
0x0070	FCCU Event Enable HV1 (FEE_HV1)	Section 54.5.20
0x0074–0x007F	Reserved	
0x0080	Supply Gauge Status Register (GR_S)	Section 54.5.21
0x0084	Pending Gauge Status Register (GR_P)	Section 54.5.22
0x0088	Global Interrupt Enable Register (IE_G)	Section 54.5.23
0x008C–0x0103	Reserved	
0x0104	Voltage Selection of IO (VSIO)	Section 54.5.24
0x0108–0x023B	Reserved	
0x023C	HPREG/DREG Selection Status Register (HPREG_DREG_SEL_STATUS)	Section 54.5.25
0x0240	PMOS/NMOS Regulator Status Register (PNREG_S)	Section 54.5.26
0x0244–0x0247	Reserved	
0x0248	Miscellaneous Ctrl Register (MISC_CTRL_REG)	Section 54.5.27
0x024C	SSWU Ctrl Register (SSWU_CTRL_REG)	Section 54.5.28
0x0250–0x02FF	Reserved	
0x0300	Temperature Event Pending register (EPR_TD)	Section 54.5.29
0x0304	Temperature Reset Event Enable register (REE_TD)	Section 54.5.30

Table 1107. Power Management Controller Memory Map (continued)

Offset	Register name	Location
0x0308	Temperature Reset Event Selection register (RES_TD)	Section 54.5.31
0x030C	Temperature Sensor Configuration Register (CTL_TD)	Section 54.5.32
0x0310–0x0317	Reserved	
0x0318	Temperature FCCU Event Enable Register (FEE_TD)	Section 54.5.33
0x031C–0x0323	Reserved	
0x0324	ESR0 Configuration Register (ESR0_CFG)	Section 54.5.34
0x0328–0x0367	Reserved	
0x0368	Digital_Regulator_monitor (DREG_MON)	Section 54.5.35
0x036C	Digital Regulator QVF Status Register (DREG_QVF_STATUS)	Section 54.5.36
0x0370	Digital Regulator Vref Status Register (DREG_VREF_STATUS)	Section 54.5.37
0x0374	DigItal Regulator Delay Status Register (DREG_DELAY_STATUS)	Section 54.5.38
0x0378	Digital Controller Slope0 (DREG_SLOPE0)	Section 54.5.39
0x037C	Digital Controller Slope1 (DREG_SLOPE1)	Section 54.5.40
0x0380	Digital Controller Slope2 (DREG_SLOPE2)	Section 54.5.41
0x0384	Digital Controller Slope3 (DREG_SLOPE3)	Section 54.5.42
0x0388	Digital Controller Slope4 (DREG_SLOPE4)	Section 54.5.43
0x038C	Digital Controller Slope5 (DREG_SLOPE5)	Section 54.5.44
0x0390	Digital Controller Slope6 (DREG_SLOPE6)	Section 54.5.45
0x0394	Digital Controller Slope7 (DREG_SLOPE7)	Section 54.5.46
0x0398	Digital Controller Slope8 (DREG_SLOPE8)	Section 54.5.47
0x039C	Digital Controller Slope9 (DREG_SLOPE9)	Section 54.5.48
0x03A0	Digital Controller Slope10 (DREG_SLOPE10)	Section 54.5.49
0x03A4–0x03CB	Reserved	
0x03CC	User BIST Flags Phase1 Register (BIST_FLAGS_PHASE1)	Section 54.5.50
0x03D0	User BIST Flags Phase2 Register (BIST_FLAGS_PHASE2)	Section 54.5.51
0x03D4	User BIST Control Register (BIST_CTRL)	Section 54.5.52
0x03D8	User BIST Time1 and Time0 Register (BIST_TIME10)	Section 54.5.53
0x03DC	User BIST Time3 and Time2 Register (BIST_TIME32)	Section 54.5.54
0x03E0	User BIST Time6 and Time5 Register (BIST_TIME65)	Section 54.5.55
0x03E4	User BIST VD Under Test Monitor Register (BIST_DEBUG)	Section 54.5.56

Table 1108 shows how the 32-bit registers for LV0, LV1, HV0, and HV1 are organized.

Table 1108. 32-bit registers LV0, LV1, HV0, HV1 arrangement table

Register class	BYTE 3[(1)] (bits [0:7])	BYTE 2 (bits [8:15])	BYTE 1[(1)] (bits [16:23])	BYTE 0[(1)] (bits [24-31])
LV0	VD3 = LVD100_FL_C_SB	VD2 = Reserved	VD1 = Reserved	VD0 = Reserved
LV1	VD7 = Reserved	VD6 = HVD134_C	VD5 = Not implemented	VD4 = Not implemented
HV0	VD11 = LVD290_AS_FL_C_IF	VD10 = Reserved	VD9 = Reserved	VD8 = Reserved
HV1	VD15 = Reserved	VD14 = LVD400_AS_IF_IM	VD13 = HVD400_IF	VD12 = Not implemented

1. Refer to *Section 54.5*.

Table 1109 shows the correspondence between VD number and VD name, where HV means a 3.3 V/5.0 V supply and LV means a 1.2 V supply.

Table 1109. VD number versus VD name table

VD Number	VD Name	VD Meaning
VD0	POR031_C	LV detector asserts when PMU supply < 0.31 V
VD1	MVD082T_C	LV detector asserts when PMU supply < 0.82 V
VD2	MVD094_C	LV detector asserts when PMU supply < 0.94 V
VD2	MVD094_FA/FB	LV detector asserts when FLASH supply < 0.94 V
VD3	LVD100_C	LV detector asserts when PMU supply < 1.0 V
VD3	LVD100_FL	LV detector asserts when FLASH supply < 1.0 V
VD3	LVD100_SB	LV detector asserts when standby supply < 1.0 V
VD6	HVD134_C	LV detector asserts when PMU supply > 1.34 V
VD7	UVD140_C	LV detector asserts when PMU supply > 1.40 V
VD7	UVD140_FL	LV detector asserts when FLASH supply > 1.40 V
VD8	POR200_C	HV detector asserts when PMU supply < 2.00 V
VD9	MVD240T_C	HV detector asserts when PMU supply < 2.40 V
VD10	MVD270_C	HV detector asserts when PMU supply < 2.70 V
VD10	MVD270_FL	HV detector asserts when FLASH supply < 2.70 V
VD11	LVD290_AS	HV detector asserts when SAR ADC supply < 2.90 V
VD11	LVD290_C	HV detector asserts when PMU supply < 2.90 V
VD11	LVD290_FL	HV detector asserts when FLASH supply < 2.90 V
VD11	LVD290_IF	HV detector asserts when IO ETHERNET supply < 2.90 V
VD13	HVD400_IF	HV detector asserts when IO ETHERNET supply > 4.00 V
VD14	LVD400_AS	HV detector asserts when SAR ADC supply < 4.00 V

Table 1109. VD number versus VD name table (continued)

VD Number	VD Name	VD Meaning
VD14	LVD400_IF	HV detector asserts when IO ETHERNET supply < 4.00 V
VD14	LVD400_IM	HV detector asserts when IO MAINS supply < 4.00 V
VD15	UVD600_FL	HV detector asserts when FLASH supply > 6.00 V
VD15	UVD600_IF	HV detector asserts when IO ETHERNET supply > 6.00 V

54.5 Registers description

54.5.1 Event Pending Register (EPR_LV0)

This Event Pending Register indicates the present and past state of the voltage detect signals in LOW VOLTAGE 0 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0000 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LVD3_SB	0	LVD3_FL	0	LVD3_C	0	0	0	0	0	0	0	0
W				w1c		w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1082. Event Pending Register (EPR_LV0)

Table 1110. EPR_LV0 field descriptions

Field	Description
3 LVD3_SB	LVD3_SB flag This bit is the low-voltage status flag associated with LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V supply always on IO ring. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 1.00 V point supply.
5 LVD3_FL	LVD3_FL flag This bit is the low-voltage status flag associated with LVD100_FL VD, the voltage level detect for the low voltage Flash 1.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 1.00 V point supply.
7 LVD3_C	LVD3_C flag This bit is the low-voltage status flag associated with LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 1.00 V point supply.

54.5.2 Reset Event Enable Register (REE_LV0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0004 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	REE3_SB	0	REE3_FL	0	REE3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1083. Reset Event Enable Register (REE_LV0)

Table 1111. REE_LV0 field descriptions

Field	Description
3 REE3_SB	REE3_SB reset enable This bit defines whether LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V always on IO ring supply, generates a system reset. The RES3_SB bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
5 REE3_FL	REE3_FL reset enable This bit defines whether LVD100_FL VD, the voltage level detect for the low voltage of 1.00 V on Flash supply, generates a system reset. The RES3_FL bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
7 REE3_C	REE3_C reset enable This bit defines whether LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply, generates a system reset. The RES3_C bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

54.5.3 Reset Event Select Register (RES_LV0)

This Reset Event Select Register controls, in LOW VOLTAGE 0 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0008 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	RES3_SB	0	RES3_FL	0	RES3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1084. Reset Event Select Register (RES_LV0)

Table 1112. RES_LV0 field descriptions

Field	Description
3 RES3_SB	RES3_SB reset event select This bit defines whether LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V always on IO ring supply of the cold point, generates a destructive reset or a functional reset. The REE3_SB bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset. **Note:** for LVD100_SB it is not allowed to enable destructive reset, only functional reset is supported.
5 RES3_FL	RES3_FL reset event select This bit defines whether LVD100_FL VD, the voltage level detect for the low voltage Flash 1.00 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_FL bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
7 RES3_C	RES3_C reset event select This bit defines whether LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_C bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

54.5.4 Interrupt Enable (IE_LV0)

This configuration register contains a set of LV0 interrupt Enable bits that correspond to each of the LV0 Event Pending Registers (EPR_LV0). These bits indicate whether an interrupt is enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0088h. The IE_LV0 bits can be read or written at any time.

Offset: 0x000C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	VD3IE_SB	0	VD3IE_FL	0	VD3IE_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1085. Interrupt Enable Register (IE_LV0)

Table 1113. IE_LV0 field descriptions

Field	Description
3 VD3IE_SB	VD3IE_SB Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V always on IO ring supply event. 0 Interrupt disabled. 1 Interrupt enabled. **Note:** Interrupt occurrence in case of LVD happens only in normal mode (interrupt being not powered in standby mode)
5 VD3IE_FL	VD3IE_FL Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD100_FL VD, the voltage level detect for the low voltage Flash 1.00 V supply event. 0 Interrupt disabled. 1 Interrupt enabled.
7 VD3IE_C	VD3IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply event. 0 Interrupt disabled. 1 Interrupt enabled.

54.5.5 FCCU Event Enable Register (FEE_LV0)

This FCCU Event Enable Register LV0 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0010 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	FEE3_SB	0	FEE3_FL	0	FEE3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1086. FCCU Event Enable Register (FEE_LV0)

Table 1114. FEE_LV0 field descriptions

Field	Description
3 FEE3_SB	FEE3_SB FCCU event enable This bit defines whether a VD assertion of LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V always on IO ring supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the always on IO ring does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the always on IO ring causes an FCCU event. **Note:** FCCU event in case of LVD happens only in normal mode (FCCU being not powered in standby mode)
5 FEE3_FL	FEE3_FL FCCU event enable This bit defines whether a VD assertion of LVD100_FL VD, the voltage level detect for the low voltage Flash 1.00 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the Flash does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the Flash causes an FCCU event.
7 FEE3_C	FEE3_C FCCU event enable This bit defines whether a VD assertion of LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the Core causes an FCCU event.

54.5.6 Event Pending Register (EPR_LV1)

This Event Pending Register indicates the present and past state of the voltage detect signals in LOW VOLTAGE 1 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a ‘1’ must be written to the flag bit that needs to be cleared.

Offset: 0x0020 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD6_C
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1087. Event Pending Register (EPR_LV1)

Table 1115. EPR_LV1 field descriptions

Field	Description
15 HVD6_C	HVD6_C flag This bit is the low-voltage status flag associated with HVD134_C VD, the voltage level detect for the low voltage Core 1.34 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 6 threshold, and clears when the supply rises above its corresponding HVD6 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 1.34 V point supply.

54.5.7 Reset Event Enable Register (REE_LV1)

This Reset Event Enable Register controls, in LOW VOLTAGE 1 range, whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0024 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	REE6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1088. Reset Event Enable Register (REE_LV1)

Table 1116. REE_LV1 field descriptions

Field	Description
15 REE6_C	REE6_C reset enable This bit defines whether a HVD134_C VD, the voltage level detect for the low voltage Core 1.34 V supply of the cold point, generates a system reset. RES_LV1[RES6_C] determines whether the reset is functional or destructive. 0Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

54.5.8 Reset Event Select Register (RES_LV1)

This Reset Event Select Register controls, in LOW VOLTAGE 1 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0028 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RES6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1089. Reset Event Select Register (RES_LV1)

Table 1117. RES_LV1 field descriptions

Field	Description
15 RES6_C	RES6_C event select This bit defines whether a HVD134_C VD, the voltage level detect for the low voltage Core 1.34 V supply of the cold point, generates a destructive reset or a functional reset. REE_LV1[REE6_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

54.5.9 Interrupt Enable (IE_LV1)

This configuration register contains a set of LV1 interrupt Enable bits that correspond to each of the LV1 Event Pending Registers (EPR_LV1). These bits indicate whether an interrupt is enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0088h. The IE_LV1 bits can be read or written at any time.

Offset: 0x002C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD6IE_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1090. Interrupt Enable Register (IE_LV1)

Table 1118. IE_LV1 field descriptions

Field	Description
15 HVD6IE_C	HVD6IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the HVD134_C VD, the voltage level detect for the low voltage Core 1.34 V supply the voltage event occurs. 0 Interrupt disabled. 1 Interrupt enabled.

54.5.10 FCCU Event Enable Register (FEE_LV1)

This FCCU Event Enable Register LV1 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0030 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1091. FCCU Event Enable Register (FEE_LV1)

Table 1119. FEE_LV1 field descriptions

Field	Description
15 FEE6_C	FEE6_C FCCU event enable This bit defines whether a HVD134_C VD assertion of Core 1.34 V supply generates an FCCU event. 0 Disabled. A VD assertion of LV1 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of LV1 on the supply of the Core causes an FCCU event.

54.5.11 Event Pending Register (EPR_HV0)

This Event Pending Register indicates the present and past state of the voltage detect signals in HIGH VOLTAGE 0 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a ‘1’ must be written to the flag bit that needs to be cleared.

Offset: 0x0040 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LVD11_AS	LVD11_IF	0	0	LVD11_FL	LVD11_C	0	0	0	0	0	0	0	0
W			w1c	w1c			w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1092. Event Pending Register (EPR_HV0)

Table 1120. EPR_HV0 field descriptions

Field	Description
2 LVD11_AS	LVD11_AS flag This bit is the high-voltage status flag associated with LVD290_AS, the voltage level detect for the low voltage SAR ADC 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 2.90 V point supply.
3 LVD11_IF	LVD11_IF flag This bit is the high-voltage status flag associated with LVD290_IF, the voltage level detect for the low voltage IO Ethernet 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 2.90 V point supply.
6 LVD11_FL	LVD11_FL flag This bit is the high-voltage status flag associated with LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 2.90 V point supply.
7 LVD11_C	LVD11_C flag This bit is the high-voltage status flag associated with LVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 2.90 V point supply.

54.5.12 Reset Event Enable Register (REE_HV0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0044 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	REE11_AS	REE11_IF	0	0	REE11_FL	REE11_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1093. Reset Event Enable Register (REE_HV0)

Table 1121. REE_HV0 field descriptions

Field	Description
2 REE11_AS	LVD11_AS reset enable This bit defines whether LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply generates a system reset. The RES11_AS bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.
3 REE11_IF	LVD11_IF reset enable This bit defines whether LVD290_IF VD, the voltage level detect for the high voltage assertion on the IO /Ethernet 2.90 V supply generates a system reset. The RES11_IF bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.
6 REE11_FL	LVD11_FL reset enable This bit defines whether LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply of the cold point generates a system reset. The RES11_FL bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
7 REE11_C	LVD11_C reset enable This bit defines whether LVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply, of the cold point generates a system reset. The RES11_C bit determines whether a system reset is enabled or not. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

54.5.13 Reset Event Select Register (RES_HV0)

This Reset Event Select Register controls, in HIGH VOLTAGE 0 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0048 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RES11_AS	RES11_IF	0	0	RES11_FL	RES11_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1094. Reset Event Select Register (RES_HV0)

Table 1122. RES_HV0 field descriptions

Field	Description
2 RES11_AS	RES11_AS reset event select This bit defines whether an LVD assertion on the SAR ADC 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_AS bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
3 RES11_IF	RES11_IF reset event select This bit defines whether an LVD assertion on the IO Ethernet 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_IF bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

Table 1122. RES_HV0 field descriptions (continued)

Field	Description
6 RES11_FL	RES11_FL reset event select This bit defines whether an LVD assertion on the Flash 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_FL bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
7 RES11_C	RES11_C reset event select This bit defines whether an LVD assertion on the Core 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_C bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

54.5.14 Interrupt Enable (IE_HV0)

This configuration register contains a set of HV0 interrupt Enable bits that correspond to each of the HV0 Event Pending Registers (EPR_HV0). These bits indicate whether an interrupt enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0088h. The IE_HV0 bits can be read or written at any time.

Offset: 0x004C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LVD11IE_AS	LVD11IE_IF	0	0	LVD11IE_FL	LVD11IE_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1095. Interrupt Enable Register (IE_HV0)

Table 1123. IE_HV0 field descriptions

Field	Description
2 LVD11IE_AS	LVD11IE_AS Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
3 LVD11IE_IF	LVD11IE_IF Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_IF VD, the voltage level detect for the high voltage assertion on the IO Ethernet 2.90 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
6 LVD11IE_FL	LVD11IE_FL Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
7 LVD11IE_C	LVD11IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the lVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.

54.5.15 FCCU Event Enable Register (FEE_HV0)

This FCCU Event Enable Register HV0 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0050 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FEE11_AS	FEE11_IF	0	0	FEE11_FL	FEE11_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1096. FCCU Event Enable Register (FEE_HV0)

Table 1124. FEE_HV0 field descriptions

Field	Description
2 FEE11_AS	FEE11_AS FCCU event enable This bit defines whether a LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the ADC does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the ADC causes an FCCU event.
3 FEE11_IF	FEE11_IF FCCU event enable This bit defines whether a LVD290_IF VD, the voltage level detect for the high voltage assertion on the IO Ethernet 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the IO Ethernet does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the IO Ethernet causes an FCCU event.
6 FEE11_FL	FEE11_FL FCCU event enable This bit defines whether a LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the Flash does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the Flash causes an FCCU event.
7 FEE11_C	FEE11_C FCCU event enable This bit defines whether a LVD290_C VD, the voltage level detect for the high voltage assertion on the Core 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the Core causes an FCCU event.

54.5.16 Event Pending Register (EPR_HV1)

This Event Pending Register indicates the present and past state of the voltage detect signals in HIGH VOLTAGE 1 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0060 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	LVD14_AS	0	LVD14_IF	0	LVD14_IM	0
W											w1c		w1c		w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	HVD13_IF	0	0	0	0	0	0	0	0	0	0
W						w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1097. Event Pending Register (EPR_HV1)

Table 1125. EPR_HV1 field descriptions

Field	Description
10 LVD14_AS	LVD14_AS flag This bit is the High voltage status flag associated with LVD400_AS VD, the VD for the high voltage SAR ADC 4.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_AS threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.
12 LVD14_IF	LVD14_IF flag This bit is the High voltage status flag associated with LVD400_IF VD, the VD for the high voltage IO Ethernet 4.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_IF threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.

Table 1125. EPR_HV1 field descriptions (continued)

Field	Description
14 LVD14_IM	LVD14_IM flag This bit is the High voltage status flag associated with LVD400_IM VD, the VD for the high voltage IO main 4.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_IM threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.
21 HVD13_IF	HVD13_IF flag This bit is the High Voltage status flag associated with HVD400_IF VD, the VD for the high voltage IO Ethernet 4.00 V supply. It is asserted when the supply is above its corresponding Voltage Detector 13 threshold, and clears when the supply falls below its corresponding HVD13_IF threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.

54.5.17 Reset Event Enable Register (REE_HV1)

This Reset Event Enable Register controls, in HIGH VOLTAGE 1 range, whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0064 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	REE14_AS	0	REE14_IF	0	REE14_IM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	REE13_IF	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1098. Reset Event Enable Register (REE_HV1)

Table 1126. REE_LV1 field descriptions

Field	Description
10 REE14_AS	REE14_AS reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES14_AS bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
12 REE14_IF	REE14_IF reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES14_IF bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
14 REE14_IM	REE14_IM reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES14_IM bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
21 REE13_IF	REE13_IF reset enable This bit defines whether HVD400_IF VD, the VD for the high voltage IO Ethernet 4.00 V supply generates a system reset. The RES13_IF bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

54.5.18 Reset Event Select Register (RES_HV1)

This Reset Event Select Register controls, in HIGH VOLTAGE 1 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0068 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	RES14_AS	0	RES14_IF	0	RES14_IM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RES13_IF	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1099. Reset Event Select Register (RES_HV1)

Table 1127. RES_HV1 field descriptions

Field	Description
10 RES14_AS	RES14_AS reset event select This bit defines whether a LVD400_AS VD, the VD for the high voltage SAR ADC 4.00 V supply generates a destructive reset or a functional reset by LVD14_AS bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
12 RES14_IF	RES14_IF reset event select This bit defines whether a LVD400_IF VD, the VD for the high voltage IO Ethernet 4.00 V supply generates a destructive reset or a functional reset by LVD14_IF bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
14 RES14_IM	RES14_IM reset event select This bit defines whether a LVD400_IM VD, the VD for the high voltage IO mains 4.00 V supply generates a destructive reset or a functional reset by LVD14_IM bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
21 RES13_IF	RES13_IF reset event select This bit defines whether a HVD400_IF VD, the VD for the high voltage IO Ethernet 4.00 V supply generates a destructive reset or a functional reset by HVD13_IF bit. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.

54.5.19 Interrupt Enable Register (IE_HV1)

This configuration register contains a set of HV1 interrupt Enable bits that correspond to each of the HV1 Event Pending Registers (EPR_HV1). These bits indicate whether an interrupt is enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0088h. The IE_HV1 bits can be read or written at any time.

Offset: 0x006C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	LVDIE14_AS	0	LVDIE14_IF	0	LVDIE14_IM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	HVDIE13_IF	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1100. Interrupt Enable Register (IE_HV1)

Table 1128. IE_HV1 field descriptions

Field	Description
10 LVDIE14_AS	LVDIE14_AS Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD400_AS VD, the VD for the high voltage SAR ADC 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
12 LVDIE14_IF	LVDIE14_IF Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD400_IF VD, the VD for the high voltage IO Ethernet 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.

Table 1128. IE_HV1 field descriptions (continued)

Field	Description
14 LVDIE14_IM	LVDIE14_IM Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD400_IM VD, the VD for the high voltage IO main 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
21 HVDIE13_IF	HVDIE13_IF Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the HVD400_IF VD, the VD for the high voltage IO Ethernet 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.

54.5.20 FCCU Event Enable Register (FEE_HV1)

This FCCU Event Enable Register HV1 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0070 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	FEE14_AS	0	FEE14_IF	0	FEE14_IM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	FEE13_IF	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1101. FCCU Event Enable Register (FEE_HV1)

Table 1129. FEE_HV1 field descriptions

Field	Description
10 FEE14_AS	FEE14_AS FCCU event enable This bit defines whether a VD assertion of LV14 on LVD400_AS for the supply of the SAR ADC generates an FCCU event. 0 Disabled. A VD assertion of LV14 does not cause an FCCU event. 1 Enabled. A VD assertion of LV14 causes an FCCU event.
12 FEE14_IF	FEE14_IF FCCU event enable This bit defines whether a VD assertion of LV14 on LVD400_IF for the supply of the IO Ethernet generates an FCCU event. 0 Disabled. A VD assertion of LV14 does not cause an FCCU event. 1 Enabled. A VD assertion of LV14 causes an FCCU event.
14 FEE14_IM	FEE14_IM FCCU event enable This bit defines whether a VD assertion of LV14 on LVD400_IM for the supply of the IO main generates an FCCU event. 0 Disabled. A VD assertion of LV14 does not cause an FCCU event. 1 Enabled. A VD assertion of LV14 causes an FCCU event.
21 FEE13_IF	FEE13_IF FCCU event enable This bit defines whether a VD assertion of HV14 on HVD400_IF for the supply of the IO Ethernet generates an FCCU event. 0 Disabled. A VD assertion of HV14 does not cause an FCCU event. 1 Enabled. A VD assertion of HV14 causes an FCCU event.

54.5.21 Supply Gauge Status Register (*GR_S*)

This status register contains the gauge of the indicated supply. This indicates the present state of the voltage detect events for each of the supplies. Each of the voltage detect signals for a given voltage are combined to form a single bit, refer to *Table 1130*.

Offset: 0x0080 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TEMP_2	TEMP_1	TEMP_0	0	0	0	0	VD14_AS	VD14_IF	0	VD14_IM	0	VD13_IF	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	VD11_AS	VD11_IF	0	VD11_FL	VD11_C	0	VD6_C	0	VD3_SB	VD3_FL	VD3_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1102. Supply Gauge Status Register (*GR_S*)

Table 1130. *GR_S* field descriptions

Field	Description
1 TEMP_2	TEMP_2 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (150 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold. 0 Currently no occurrence. 1 Temperature occurrence detected.
2 TEMP_1	TEMP_1 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (135 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold. 0 Currently no occurrence. 1 Temperature occurrence detected.
3 TEMP_0	TEMP_0 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (-40 °C). It is asserted when the temperature falls below its corresponding threshold, and clears when the temperature is above its corresponding threshold. 0 Currently no occurrence. 1 Temperature occurrence detected.
8 VD14_AS	VD14_AS low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD400_AS for the high voltage 4.00 V SAR ADC supply. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies are above the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 4.00 V supplies.
9 VD14_IF	VD14_IF low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD400_IF for the high voltage 4.00 V IO Ethernet supply. It is asserted when the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies are above the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 4.00 V supplies.
11 VD14_IM	VD14_IM low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD400_IM for the high voltage 4.00 V IO main supply. It is asserted when the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies fall below the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 4.00 V supplies.

Table 1130. *GR_S* field descriptions (continued)

Field	Description
13 VD13_IF	VD13_IF high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect HVD400_IF for the high voltage 4.00 V IO Ethernet supply. It is asserted when the IO supply rises above its corresponding HVD threshold, and clears when the supply rises above its corresponding HVD threshold. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 4.00 V supply.
21 VD11_AS	VD11_AS low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_AS for the high voltage 2.90 V SAR ADC supplies. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.90 V supplies.
22 VD11_IF	VD11_IF low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_IF for the high voltage 2.90 V IO Ethernet supplies. It is asserted when any of the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.90 V supplies.
24 VD11_FL	VD11_FL low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_FL for the high voltage 2.90 V Flash supplies. It is asserted when any of the Flash supplies falls below the corresponding LVD threshold, and clears when all the Flash supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.90 V supplies.
25 VD11_C	VD11_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_C for the high voltage 2.90 V Core supplies. It is asserted when any of the Core supplies falls below the corresponding LVD threshold, and clears when all the Core supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.90 V supplies.
27 VD6_C	VD6_C high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect HVD134_C for the high voltage 1.34 V Core supplies. It is asserted when any of the Core supplies rises above the corresponding HVD threshold, and clears when all the Core supplies fall below the corresponding HVD threshold. All the 1.34 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 1.34 V supply.

Table 1130. *GR_S* field descriptions (continued)

Field	Description
29 VD3_SB	VD3_SB low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD100_SB for the low voltage 1.00 V always on supplies. It is asserted when any of the always on ring supplies falls below the corresponding LVD threshold, and clears when the always on supply rises above the corresponding LVD threshold. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 1.00 V supplies.
30 VD3_FL	VD3_FL low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD100_FL for the high voltage 1.00 V Flash supplies. It is asserted when any of the Flash supplies falls below the corresponding LVD threshold, and clears when all the Flash supplies rise above the corresponding LVD threshold. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 1.00 V supplies.
31 VD3_C	VD3_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD100_C for the low voltage 1.00 V Core supplies. It is asserted when any of the Core supplies falls below the corresponding LVD threshold, and clears when all the Core supplies rise above the corresponding LVD threshold. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 1.00 V supplies.

54.5.22 Pending Gauge Status Register (GR_P)

This configuration register contains a mirror of the contents of the Event Pending Registers, but in a format that includes one bit for each voltage detect signal. These bits indicate the state of the voltage detect events including whether a voltage level event has occurred at any time in the past but has not been cleared (via the EPR register). These bits are read only, and they are cleared by writing a one to the corresponding EPR register.

The Pending registers are only set on an active edge of an event, so a transition must be seen before they will become set.

The raw analog block voltage detect signals are also synchronized to the bus clock before being used for the pending registers.

Offset: 0x0084 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VD14_AS	VD14_IF	0	VD14_IM	0	VD13_IF	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	VD11_AS	VD11_IF	0	VD11_FL	VD11_C	0	VD6_C	0	VD3_SB	VD3_FL	VD3_C	0	IRQ_ST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1103. Pending Gauge Status Register (GR_P)

Table 1131. GR_P field descriptions

Field	Description
8 VD14_AS	VD14_as Pending Monitor. Read-only register.
9 VD14_IF	VD14_if Pending Monitor. Read-only register.
11 VD14_IM	VD14_im Pending Monitor. Read-only register.
13 VD13_IF	VD13_if Pending Monitor. Read-only register.
18 VD11_AS	VD11_as Pending Monitor. Read-only register.
19 VD11_IF	VD11_IF Pending Monitor. Read-only register.
21 VD11_FL	VD11_FL Pending Monitor. Read-only register.
22 VD11_C	VD11_C Pending Monitor. Read-only register.
24 VD6_C	VD6_C Pending Monitor. Read-only register.
26 VD3_SB	VD3_SB Pending Monitor. Read-only register.
27 VD3_FL	VD3_FL Pending Monitor. Read-only register.

Table 1131. GR_P field descriptions (continued)

Field	Description
28 VD3_C	VD3_C Pending Monitor. Read-only register.
30 IRQ_ST	IRQ_ST user BIST Pending Monitor. Read-only register.

54.5.23 Interrupt Enable Pending Register (IE_G)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Event Pending Registers (EPR_XX), but in a format similar to the Pending Gauge Status register (GR_P). These bits indicate whether an interrupt is enabled. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are enabled via a write of ‘1’ to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

Offset: 0x0088 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IE_EN	0	0	0	0	0	0	0	VD14IE_AS	VD14IE_IF	0	VD14IE_IM	0	VD13IE_IF	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	VD11IE_AS	VD11IE_IF	0	VD11IE_FL	VD11IE_C	0	VD6IE_C	0	VD3IE_SB	VD3IE_FL	VD3IE_C	0	IRQ_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1104. Interrupt Enable Global Register (IE_G)

Table 1132. IE_G field descriptions

Field	Description
0 IE_EN	IE_EN. Interrupt Enable This bit permits Interrupt Enable bit to be written. 0 No interrupt enables can be written. 1 Any interrupt enable can be written.
8 VD14IE_AS	VDIE14_AS enable interrupt monitor. Read-only register.
9 VD14IE_IF	VDIE14_IF enable interrupt monitor. Read-only register.
11 VD14IE_IM	VDIE14_IM enable interrupt monitor. Read-only register.

Table 1132. IE_G field descriptions (continued)

Field	Description
13 VD13IE_IF	VDIE13_IF enable interrupt monitor. Read-only register.
18 VD11IE_AS	VDIE11_AS enable interrupt monitor. Read-only register.
19 VD11IE_IF	VDIE11_IF enable interrupt monitor. Read-only register.
21 VD11IE_FL	VDIE11_FL enable interrupt monitor. Read-only register.
22 VD11IE_C	VDIE11_C enable interrupt monitor. Read-only register.
24 VD6IE_C	VDIE6_C enable interrupt monitor. Read-only register.
26 VD3IE_SB	VDIE3_SB enable interrupt monitor. Read-only register.
27 VD3IE_FL	VDIE3_FL enable interrupt monitor. Read-only register.
28 VD3IE_C	VDIE3_C enable interrupt monitor. Read-only register.
30 IRQ_EN	IRQ_EN user BIST enable interrupt monitor. Read-only register.

54.5.24 Voltage selection of IO (VSIO)

This register controls the pad operation in the different I/O supply segments. The configuration bits have to be set accordingly to the supply voltage applied to the I/O segments in order to get the correct behavior of the input and output buffers.

Offset: 0x0104 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	VSIO_IM	VSIO_IF	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 1105. Voltage selection of IO Register (VSIO)

Table 1133. VSIO field description

Field	Description
29 VSIO_IM	Selects between 3.3/5 V for IO main ring. 0 Ring will operate at 3.3 V 1 Ring will operate at 5 V
30 VSIO_IF	Selects between 3.3/5 V for IO Ethernet ring. 0 Ring will operate at 3.3 V 1 Ring will operate at 5 V

54.5.25 HPREG/DREG Selection Status Register (HPREG_DREG_SEL_STATUS)

This register indicates which internal regulator is enabled (HPREG with external ballast or DREG with internal ballast).

Offset: 0x023C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	HPREG_ENB	DREG_ENB	HPREG_MODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 1106. HPREG/DREG Selection Status Register (HPREG_DREG_SEL_STATUS)

Table 1134. HPREG_DREG_SEL_STATUS field descriptions

Field	Description
29 HPREG_ENB	This bit indicates the enable/disable status of HPREG. 0 HPREG is enabled, provided hpreg_mode = 1. 1 HPREG is disabled
30 DREG_ENB	This bit indicates the enable/disable status of DREG. 0 DREG is enabled, provided hpreg_mode = 0. 1 DREG is disabled
31 HPREG_MODE	This bit indicates the HPREG/DREG mode. 0 DREG mode 1 HPREG mode

54.5.26 PMOS/NMOS Regulator Status Register (PNREG_S)

This register shows the status of PMOS regulator and NMOS regulator. It also indicates internal or external regulator mode.

Offset: 0x0240 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PREG_ENB	NREG_ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1107. PMOS/NMOS Regulator Status Register (PNREG_S)

Table 1135. PNREG_S field descriptions

Field	Description
30 PREG_ENB	This bit is used to enable/disable PREG in HPREG mode 0 PREG is enabled 1 PREG is off PREG is always enabled in DREG mode
31 NREG_ENB	This bit is used to enable/disable NREG in HPREG mode 0 NREG is enabled 1 NREG regulator is off NREG is always enabled in DREG mode

54.5.27 Miscellaneous CTRL REG (MISC_CTRL_REG)

This register provides the controlling of both LPRC regulator and oscillator.

Offset: 0x0248 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LPRCREG_ENB(1)	1	0	RESERVED(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

1. Software must keep this bit to 0

Figure 1108. Miscellaneous CTRL REG (MISC_CTRL_REG)

Table 1136. MISC_CTRL_REG field descriptions

Field	Description
28 LPRCREG_ENB	Enabling/Disabling of LPRC regulator 0 regulator enabled 1 regulator disabled

54.5.28 SSWU CTRL REG (SSWU_CTRL_REG)

This register provides the controlling of SSWU enable and its prescaler programming value.

Offset: 0x024C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	SSWU_PRESCALER						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1109. SSWU CTRL REG(SSWU_CTRL_REG)

Table 1137. SSWU_CTRL_REG field descriptions

Field	Description
25:31 SSWU_PRESCALER	Prescaler value for SSWU. **Note:** The resulting TU prescaler will be SSWU_PRESCALER + 1. Refer to *Section 11.6.6: TU*

54.5.29 Event Pending Register (EPR_TD)

This Event Pending Register indicates the present and past state of the temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a ‘1’ must be written to the flag bit that needs to be cleared.

Offset: 0x0300 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1110. Event Pending Register (EPR_TD)

Table 1138. EPR_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (150 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a ‘1’ is written to this bit location. *If the IETD_2 bit is also asserted, an interrupt is sent to the CPU*. If REE_TD[TEMP_2] is also asserted, a system reset is generated, *which clears IETD_2 and negate the interrupt request*. If REE_TD[TEMP_2] is asserted and a destructive reset is selected (via RES_TD[TEMP_2] being 0), then a destructive reset is generated. If RES_TD[TEMP_2] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected
30 TEMP_1	TEMP_1 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (135 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a ‘1’ is written to this bit location. *If the IETD_1 bit is also asserted, an interrupt is sent to the CPU*. If REE_TD[TEMP_1] is also asserted, a system reset is generated, *which clears IETD_1 and negate the interrupt request*. If REE_TD[TEMP_1] is asserted and a destructive reset is selected (via RES_TD[TEMP_1] being 0), then a destructive reset is generated. If RES_TD[TEMP_1] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected
31 TEMP_0	TEMP_0 flag This bit is the temperature status flag associated with the temperature for the *cold* temperature sensor point. It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a ‘1’ is written to this bit location. *If the IETD_0 bit is also asserted, an interrupt is sent to the CPU*. If REE_TD[TEMP_0] is also asserted, a system reset is generated, *which clears IETD_0 and negate the interrupt request*. If REE_TD[TEMP_0] is asserted and a destructive reset is selected (via RES_TD[TEMP_0] being 0), then a destructive reset is generated. If RES_TD[TEMP_0] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.

54.5.30 Reset Event Enable Register (REE_TD)

This Reset Event Enable Register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Offset: 0x0304 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1111. Reset Event Enable Register (REE_TD)

Table 1139. REE_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 reset enable This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset 1 Enabled. Temperature Sensor detect causes system reset
30 TEMP_1	TEMP_1 reset enable This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_1] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset 1 Enabled. Temperature Sensor detect causes system reset
31 TEMP_0	TEMP_0 reset enable This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.

54.5.31 Reset Event Select Register (RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the temperature sensor event causes either a destructive or functional reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Offset: 0x0308 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1112. Reset Event Select Register (RES_TD)

Table 1140. RES_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 reset event select This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_2] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
30 TEMP_1	TEMP_1 reset event select This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_1] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
31 TEMP_0	TEMP_0 reset event select This bit defines whether an temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_0] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.

54.5.32 Temperature Sensor Configuration Register (CTL_TD)

The Temperature Sensor Configuration Registers (CTL_TD) contains two enables that must both be enabled in order for the Temperature Sensor to be operable. It also includes two 5-bit signed trim adjustment register fields that are used to modify both TS trim values (one bus for over temperature and one bus for under temperature).

Offset: 0x030C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	PMC_AOUT_EN	TRIM_ADJ_OVER1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS2IE	TS1IE	TS0IE	0	TRIM_ADJ_OVER				0	0	TRIM_ADJ_UNDER				AMUX_CTAT_TS_CTRL	ADC_AOUT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1113. Temperature Sensor Configuration Register (CTL_TD)

Table 1141. CTL_TD field descriptions

Field	Description
11 PMC_AOUT_EN	Temperature Sensor Enable for PMU Refer to the Temperature Sensor chapter for details.
12:15 TRIM_ADJ_OVER1	TRIM_ADJ_OVER1[3:0] Customer adjustable over trim register for PMU These bits are a signed binary number that is added to the over temperature trim (NT_TD1) value. The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
16 TS2IE	Temperature Sensor input 2 Interrupt Enable This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
17 TS1IE	Temperature Sensor input 1 Interrupt Enable This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.

Table 1141. CTL_TD field descriptions (continued)

Field	Description
18 TS0IE	Temperature Sensor input 0 Interrupt Enable This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs
20:23 TRIM_ADJ_OVER	TRIM_ADJ_OVER[3:0] Customer adjustable over trim register for ADC These bits are a signed binary number that is added to the over temperature trim (NT_TD2) value. The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
26:29 TRIM_ADJ_UNDER	TRIM_ADJ_UNDER[3:0] Customer adjustable under trim register These bits are a signed binary number that is added to the Under temperature trim value (NT_TD0). The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
30 AMUX_CTAT_TS_CTRL	Analog mux enable Refer to the Temperature Sensor chapter for details.
31 ADC_AOUT_EN	Temperature Sensor Enable for ADC Refer to the Temperature Sensor chapter for details.

54.5.33 Temperature Sensor FCCU Event Enable Register (FEE_TD)

This Temperature Sensor FCCU Event Enable Register controls whether the temperature sensor detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit (EPR_TD) is enabled, the temperature sensor detect assertion causes an FCCU event to be generated when the selected temperature exceeds the desired value.

Offset: 0x0318 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE_TS2	FEE_TS1	FEE_TS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 1114. Temperature Sensor FCCU Event Enable Register (FEE_TD)

Table 1142. Temperature Sensor FEE_TD field descriptions

Field	Description
29 FEE_TS2	Temp Sensor FCCU event enable This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
30 FEE_TS1	Temp Sensor FCCU event enable This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
31 FEE_TS0	Temp Sensor FCCU event enable This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.

54.5.34 ESR0 Configuration Register (ESR0_CFG)

The ESR0 Configuration Register (ESR0_CFG) is used to map device reset status on an external Pad. The register reset is ipg_hard_async_reset_por_b.

Offset: 0x0324 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ESR0_CFG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1115. ESR0 Configuration Register (ESR0_CFG)

Table 1143. ESR0_CFG field description

Field	Description
31 ESR0_CFG	The ESR0_CFG register is a read only register by IPS bus, and read-write register through DCF bus. 0 means the device reset status is not available on an external pad 1 means the device reset status is available on an external pad

54.5.35 Digital Regulator Monitor Register (DREG_MON)

This register indicates all important parameters of DREG controller.

Offset: 0x0368 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	QVF_PULSE	VALID	0	DELTA		0	PCURR_COMP					NCURR_COMP				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	QVF_PEAK								ACTIVE_SOURCES							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1116. Digital Regulator Monitor Register (DREG_MON)

Table 1144. DREG_MON field description

Field	Description
0 QVF_PULSE	Error flag indicator 0 No error 1 Error
1 VALID	Indicates valid thermometric code 0 thermometric codes are not valid. 1 thermometric codes are valid.
3:4 DELTA	Indicates value in 2’complement. +1 represents increase in current sources and -1 decrease in current sources
6:10 PCURR_COMP	represents PREG current computational
11:15 NCURR_COMP	represents NREG current computational
16:23 QVF_PEAK	indicates consecutive number of invalid thermometric codes
24:31 ACTIVE_SOURCES	indicates number of digital ballast currently active

54.5.36 Digital Regulator QVF Status Register (DREG_QVF_STATUS)

This register indicates the quantizer value fault status of digital regulator controller.

Offset: 0x036C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	QVF_MASK	QVF_FAULT	0	0	0	0	0	0	QVF_COUNT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1117. Digital Regulator QVF Status Register (DREG_QVF_STATUS)

Table 1145. DREG_QVF_STATUS field description

Field	Description
16 QVF_MASK	To mask QVF fault indication to the FCCU. 0 no masking of fault 1 fault is masked
17 QVF_FAULT	indicates QVF fault. 0 no fault has not occurred 1 fault has occurred
24:31 QVF_COUNT	Determines number of consecutive faults occurrence in order to raise fault flag. Programming 0x00 disables error flag.

54.5.37 Digital Regulator Vref Status Register (DREG_VREF_STATUS)

Offset: 0x0370 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DREG_DITHER	0	0	0	0	0	0	0	0	0	0	0	0	0	VREFN_LOOP_SIGN	VREFN_SATEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VREFN_INTGAIN															
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 1118. Digital Regulator Vref Status Register (DREG_VREF_STATUS)

Table 1146. DREG_VREF_STATUS field description

Field	Description
0 DREG_DITHER	To control vrefn dithering function 0 dithering disabled 1 dithering enabled
14 VREFN_LOOP_SIGN	Defines sign of vrefn control loop 0 loop in feedback direction 1 loop in reversed feedback direction
16:31 VREFN_INTGAIN	Determines vrefn control loop gain. Values in the range 32–1024 are acceptable. If programmed 0x0000 loop is disabled

54.5.38 Digital Regulator Delay Status Register (DREG_DELAY_STATUS)

This register is used for programming of settling time for digital regulator controller, digital regulator (analog) and for self detection mode.

Offset: 0x0374 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	DELAY2_SEL	DELAY2					DELAY1			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

Figure 1119. DigItal Regulator Delay Status Register (DREG_DELAY_STATUS)

Table 1147. DREG_DELAY_STATUS field description

Field	Description
22 DELAY2_SEL	The delay2_sel is used for selection between fix delay and self-detection mechanism for digital controller settling. 0 no self detection mechanism, there will be fixed delay corresponding to delay2 provided for DREG settling 1 self detection mechanism for settling of DREG controller
23:27 DELAY2	The delay2 is used for programming time from 2 µs to 80 µs for DREG controller settling in steps of 2.5 µs. Default value corresponds to 25 µs.
28:31 DELAY1	The delay1 is used for programming time 0.15625 µs to 2.34375 µs for DREG analog settling in steps of 0.15625 µs. Default value corresponds to 0 µs.

54.5.39 Digital Controller Slope0 (DREG_SLOPE0)

All the slope register (DREG_SLOPE0–10) combines to provide gain of digital ballast regulation loop.

Offset: 0x0378 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	SLOPE0															
Reset	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	0

Figure 1120. Digital Controller Slope0 (DREG_SLOPE0)

Table 1148. DREG_SLOPE0 field description

Field	Description
16:31 SLOPE0	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.40 Digital Controller Slope1 (DREG_SLOPE1)

Offset: 0x037C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	SLOPE1															
Reset	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	0

Figure 1121. Digital Controller Slope1 (DREG_SLOPE1)

Table 1149. DREG_SLOPE1 field description

Field	Description
16:31 SLOPE1	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.41 Digital Controller Slope2 (DREG_SLOPE2)

Offset: 0x0380 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	SLOPE2															
Reset	1	1	1	0	0	0	1	1	0	0	1	1	0	0	0	0

Figure 1122. Digital Controller Slope2 (DREG_SLOPE2)

Table 1150. DREG_SLOPE2 field description

Field	Description
16:31 SLOPE2	These values are in 2's complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.42 Digital Controller Slope3 (DREG_SLOPE3)

Offset: 0x0384 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	SLOPE3															
Reset	1	1	1	0	1	1	0	0	1	1	0	0	1	0	1	1

Figure 1123. Digital Controller Slope3 (DREG_SLOPE3)

Table 1151. DREG_SLOPE3 field description

Field	Description
16:31 SLOPE3	These values are in 2's complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.43 Digital Controller Slope4 (DREG_SLOPE4)

Offset: 0x0388 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLOPE4															
Reset	1	1	1	1	0	1	1	0	0	1	1	0	0	1	0	1

Figure 1124. Digital Controller Slope4 (DREG_SLOPE4)

Table 1152. DREG_SLOPE4 field description

Field	Description
16:31 SLOPE4	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.44 Digital Controller Slope5 (DREG_SLOPE5)

Offset: 0x038C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLOPE5															
Reset	1	1	1	1	1	0	1	1	0	0	1	1	0	0	1	1

Figure 1125. Digital Controller Slope5 (DREG_SLOPE5)

Table 1153. DREG_SLOPE5 field description

Field	Description
16:31 SLOPE5	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.45 Digital Controller Slope6 (DREG_SLOPE6)

Offset: 0x0390 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	SLOPE6															
Reset	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	1

Figure 1126. Digital Controller Slope6 (DREG_SLOPE6)

Table 1154. DREG_SLOPE6 field description

Field	Description
16:31 SLOPE6	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.46 Digital Controller Slope7 (DREG_SLOPE7)

Offset: 0x0394 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	SLOPE7															
Reset	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1

Figure 1127. Digital Controller Slope7 (DREG_SLOPE7)

Table 1155. DREG_SLOPE7 field description

Field	Description
16:31 SLOPE7	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.47 Digital Controller Slope8 (DREG_SLOPE8)

Offset: 0x0398 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	SLOPE8															
Reset	0	0	0	1	0	0	1	1	0	0	1	1	0	1	0	1

Figure 1128. Digital Controller Slope8 (DREG_SLOPE8)

Table 1156. DREG_SLOPE8 field description

Field	Description
16:31 SLOPE8	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.48 Digital Controller Slope9 (DREG_SLOPE9)

Offset: 0x039C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	SLOPE9															
Reset	0	0	1	0	0	1	1	0	0	1	1	0	1	0	1	0

Figure 1129. Digital Controller Slope9 (DREG_SLOPE9)

Table 1157. DREG_SLOPE9 field description

Field	Description
16:31 SLOPE9	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.49 Digital Controller Slope10 (DREG_SLOPE10)

Offset: 0x03A0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R / W	SLOPE10															
Reset	0	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1

Figure 1130. Digital Controller Slope10 (DREG_SLOPE10)

Table 1158. DREG_SLOPE10 field description

Field	Description
16:31 SLOPE10	These values are in 2’s complement. It along with other slope values provides gain for digital ballast regulation loop.

54.5.50 User BIST Flags Phase1 Register (BIST_FLAGS_PHASE1)

This configuration register contains a monitor of the flags phase1 coming from the User BIST sub-block.

Offset: 0x03CC Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	FLAGS_P1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLAGS_P1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1131. User BIST Flags Phase1 Register (BIST_FLAGS_PHASE1)

Table 1159. BIST_FLAGS_PHASE1 field description

Field	Description
10:31 FLAGS_P1	The 22 bits of field FLAGS_P1 are the phase one flags of the User BIST that indicate the status of monitor when it is tripped: – Bit 10 - FLAGS_P1[21]: UVD600B_IF – Bit 11 - FLAGS_P1[20]: UVD600B_FL – Bit 12 - FLAGS_P1[19]: HVD400B_IF – Bit 13 - FLAGS_P1[18]: UVD140B_FL – Bit 14 - FLAGS_P1[17]: UVD140B_C – Bit 15 - FLAGS_P1[16]: HVD134B_C – Bit 16 - FLAGS_P1[15]: LVD400B_IF – Bit 17 - FLAGS_P1[14]: LVD400B_IM – Bit 18 - FLAGS_P1[13]: LVD400B_AS – Bit 19 - FLAGS_P1[12]: LVD290B_IF – Bit 20 - FLAGS_P1[11]: LVD290B_AS – Bit 21 - FLAGS_P1[10]: LVD290B_FL – Bit 22 - FLAGS_P1[9]: LVD290B_C – Bit 23 - FLAGS_P1[8]: MVD270B_FL – Bit 24 - FLAGS_P1[7]: MVD270B_SB – Bit 25 - FLAGS_P1[6]: MVD270B_C – Bit 26 - FLAGS_P1[5]: LVD100B_FL – Bit 27 - FLAGS_P1[4]: LVD100B_C – Bit 28 - FLAGS_P1[3]: MVD094B_FB – Bit 29 - FLAGS_P1[2]: MVD094B_FA – Bit 30 - FLAGS_P1[1]: MVD094B_C – Bit 31 - FLAGS_P1[0]: LVD100B_SB They are read only bits.

54.5.51 User BIST Flags Phase2 Register (BIST_FLAGS_PHASE2)

This configuration register contains a monitor of the flags phase2 coming from the User BIST sub-block.

Offset: 0x03D0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	FLAGS_P2					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLAGS_P2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1132. User BIST Flags Phase2 Register (BIST_FLAGS_PHASE2)

Table 1160. BIST_FLAGS_PHASE2 field description

Field	Description
10:31 FLAGS_P2	The 22 bits of field FLAGS_P2 are the phase one flags of the User BIST that indicate the status of monitor when it is tripped: – Bit 10 - FLAGS_P1[21]: UVD600B_IF – Bit 11 - FLAGS_P1[20]: UVD600B_FL – Bit 12 - FLAGS_P1[19]: HVD400B_IF – Bit 13 - FLAGS_P1[18]: UVD140B_FL – Bit 14 - FLAGS_P1[17]: UVD140B_C – Bit 15 - FLAGS_P1[16]: HVD134B_C – Bit 16 - FLAGS_P1[15]: LVD400B_IF – Bit 17 - FLAGS_P1[14]: LVD400B_IM – Bit 18 - FLAGS_P1[13]: LVD400B_AS – Bit 19 - FLAGS_P1[12]: LVD290B_IF – Bit 20 - FLAGS_P1[11]: LVD290B_AS – Bit 21 - FLAGS_P1[10]: LVD290B_FL – Bit 22 - FLAGS_P1[9]: LVD290B_C – Bit 23 - FLAGS_P1[8]: MVD270B_FL – Bit 24 - FLAGS_P1[7]: MVD270B_SB – Bit 25 - FLAGS_P1[6]: MVD270B_C – Bit 26 - FLAGS_P1[5]: LVD100B_FL – Bit 27 - FLAGS_P1[4]: LVD100B_C – Bit 28 - FLAGS_P1[3]: MVD094B_FB – Bit 29 - FLAGS_P1[2]: MVD094B_FA – Bit 30 - FLAGS_P1[1]: MVD094B_C – Bit 31 - FLAGS_P1[0]: LVD100B_SB They are read only bits.

54.5.52 User BIST Control Register (BIST_CTRL)

The User BIST Control register (BIST_CTRL) contains both control bits and status bits to manage the User BIST.

Offset: 0x03D4 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	IRQST	0	0	0	IRQEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	NCFST				NCFEN	0	STATUS			0	0	0	START
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1133. User BIST Control Register (BIST_CTRL)

Table 1161. BIST_CTRL field descriptions

Field	Description
11 IRQST	IRQST is the InteRrupt reQuest pending STatus bit to generate a User BIST Interrupt. It is a R/W0 register. That means set by HW and cleared by SW. The destructive reset resets this bit. 0 No interrupt occurred. 1 An interrupt occurred.
15 IRQEN	IRQEN is the InteRrupt reQuest ENable bit to permit a User BIST Interrupt. It is a R/W register. The destructive reset resets this bit. 0 Interrupt is disabled. 1 Interrupt enabled.
19 NCFST	NCFST is the Not Critical Fault bit. The assertion of this bit will generate FCCU event once user BIST fails. It is set by HW and cleared by SW. The destructive reset resets this bit. 0 No User BIST NCF occurred. 1 A User BIST NCF occurred.
23 NCFEN	NCFEN is the Not Critical Fault ENable bit to permit a User BIST Not Critical Fault and Fault indication is sent to FCCU. It is a R/W register. The destructive reset resets these bits. 0 No User BIST NCF enabled. 1 A User BIST NCF enabled.
25:27 STATUS	STATUS Register are three read-only bits, that according to the value has a own meaning. 000 BIST_IDLE 001 BIST_RUN 010 BIST_PASSED 011 BIST_FAILED 100 BIST_ABORT 101 Reserved 110 Reserved 111 Reserved
31 START	START User BIST bit If set by SW a 1 is present in the START bit for a System clock cycle only, then the START bit register is clear by HW. The destructive reset resets this bit. 0 No start pulse. 1 A start pulse occurs of a 1 system clock cycle duration

54.5.53 User BIST Time1 and Time0 Register (BIST_TIME10)

The User BIST Time1 and Time0 Register (BIST_TIME10) contains two time numbers Time1 and Time0 giving the System clock cycles number required by the User BIST timer1 and timer0.

Offset: 0x03D8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TIME_1											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TIME_0											
W																
Reset	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1

Figure 1134. User BIST Time1 and Time0 Register (BIST_TIME10)

Table 1162. BIST_TIME10 field descriptions

Field	Description
4:15 TIME_1	TIME_1 is the system clock cycles number required by the TIMER 1 in the User BIST management. The destructive reset resets these bits.
20:31 TIME_0	TIME_0 is the system clock cycles number required by the TIMER 0 in the User BIST management. The destructive reset resets these bits.

54.5.54 User BIST Time3 and Time2 Register (BIST_TIME32)

The User BIST Time3 and Time2 Register (BIST_TIME32) contains two time numbers Time3 and Time2 giving the system clock cycles number required by the User BIST timer3 and timer2.

Offset: 0x03DC Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TIME_3											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TIME_2											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 1135. User BIST Time3 and Time2 Register (BIST_TIME32)

Table 1163. BIST_TIME32 field descriptions

Field	Description
4:15 TIME_3	TIME_3 is the system clock cycles number required by the TIMER 3 in the User BIST management. The destructive reset resets these bits.
20:31 TIME_2	TIME_2 is the system clock cycles number required by the TIMER 2 in the User BIST management. The destructive reset resets these bits.

54.5.55 User BIST Time6 and Time5 Register (BIST_TIME65)

The User BIST Time6 and Time5 Register (BIST_TIME65) contains two time numbers Time6 and Time5 giving the system clock cycles number required by the User BIST timer6 and timer5.

Offset: 0x03E0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TIME_6											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TIME_5											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Figure 1136. User BIST Time6 and Time5 Register (BIST_TIME65)

Table 1164. BIST_TIME65 field descriptions

Field	Description
4:15 TIME_6	TIME_6 is the system clock cycles number required by the TIMER 6 in the User BIST management. The destructive reset resets these bits.
20:31 TIME_5	TIME_5 is the system clock cycles number required by the TIMER 5 in the User BIST management. The destructive reset resets these bits.

54.5.56 User BIST VD Under Test Monitor Register (BIST_DEBUG)

This VD Under Test Monitor register contains a monitor of the Voltage Detector Under Test of the USER BIST sub-block.

Note: The BIST_DEBUG register can only be read while BIST is running. Once BIST is finished the status can be obtained by checking the BIST_FLAGS_PHASE1 and BIST_FLAGS_PHASE2 registers.

Offset: 0x03E4 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	VD_MON					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1137. User BIST VD Under Test Monitor Register (BIST_DEBUG)

Table 1165. BIST_DEBUG field description

Field	Description
26:31 VD_MON	The six read only bits of field VD_MON give the current VD Under Test of the User BIST unit among the following VDs: 0x1F LVD100_SB 0x1E MVD094_C 0x1D MVD094_FA 0x1C MVD094_FB 0x1B LVD100_C 0x1A LVD100_FL 0x15 MVD270_C 0x16 MVD270_SB 0x14 MVD270_FL 0x13 LVD290_C 0x12 LVD290_FL 0x10 LVD290_AS 0x0E LVD290_IF 0x09 LVD400_AS 0x08 LVD400_IM 0x06 LVD400_IF 0x19 HVD134_C 0x18 UVD140_C 0x17 UVD140_FL 0x0B HVD400_IF 0x04 UVD600_FL 0x02 UVD600_IF 0x00 No VD under Test

54.6 Standby entry/exit

Standby entry is an application initiated mode.The standby entry process starts once the application configures [MC_ME_MCTL] register for target mode as standby. For detailed description of entry/exit, please refer to MC_PCU.

54.7 Temperature Sensor Interface Logic

The Temperature Sensor interface consists of an analog enable, a set of mode registers, and an over-temperature threshold select control.

For more information regarding the Temperature Sensor function, please refer to the *Chapter 40: Temperature Sensor*.

54.8 Power On Reset (MC_RGM Phase Gates)

The PMU digital logic interfaces to the Reset Generation Module in order to indicate the type and conditions of the resets from the PMU and Temperature Sensor Analog blocks.

There are two signals for both the PMU and Temp Sensor blocks; one is used to indicate a destructive reset, and one is used to indicate a functional reset.

The PMU digital logic also indicates when the RGM allows transition to the next POR phase.

The digital PMU also uses signals from the RGM to indicate when some of the reset signals should be masked during the POR sequence.

During the POR phase 3 exit, the PMU digital logic uses all the enabled Voltage Detects to determine if it is correct to exit from Phase 3. Once all the enabled LVDs and HVDs have become deasserted and the time from the LVD circuits to adjust to new trim values has elapsed, then the signal is sent to the RGM that the correct voltage levels have been achieved to exit from Phase 3.

54.9 Flash interface (DCF client usage)

The PMU digital is updated during POR from the Flash block via a DCF client. This load is used for several pieces of data, but most of the data is trim values being used for analog modules (PMU, Temperature Sensor, and ADC Bandgap).

These trim bits that are loaded from the Flash are not readable via the IP bus. These trim values loaded via the DCF client are not compacted (they are written individually, and not several at a time).

A DCF client is used to interface to the Flash for these early data accesses. This client also performs triple voting to keep the data safe from stray bit flips.

Another set of data that is loaded during the DCF single read process are updated values for the REE and RES bits. Refer to *Section 54.5.30: Reset Event Enable Register (REE_TD)* for more information regarding the REE/RES bits load from Flash.

If the REE bit is loaded via the DCF as set (enabled), then it is not possible to clear this REE bit. The reset value of the REE DCF bits are all low (thus to allow the bits to be rewritable) and same is the reset value of the REE register.

The monitors for which reactions are enabled will be programmed in the corresponding bit flag of Reset Event Enable (REE_XX) register as “1” (reset enabled) and in the Reset Event Select (RES_XX) register as “0” (destructive reset selected).

The monitors for which reactions are not enabled will be programmed in the corresponding bit flag of Reset Event Enable (REE_XX) register as “0” (reset disabled) and in the Reset Event Select (RES_XX) register as “1” (functional reset selected).

The REE DCF bits loaded are compacted into a single DCF write (they do not match the IPS Bus register bit locations). The REE DCF loaded bits do not have the same limitation that they can only be written once and any further writes must be to the same value. The REE DCF bits can be rewritten to a new value during a new reset event.

When the REE DCF bits are updated, the REE and RES register bits are updated at the same time to the new value.

All of the DCF clients are reset via the POR reset signal.

For more information regarding the DCF records, please refer to the DCF Records chapter.

The trim data is read out of the Flash’s differential read space (slower access, but less issues with voltage variances). Once the differential reads are completed, the Flash indicates this to the PMU digital via the SSCM module.

Once the PMU trim values have been read, the analog circuit has been given time to update to the new trim values, and the LVDs have all deasserted, a signal is asserted to the Flash to indicate that the voltage is high enough for faster single access reads.

All of the LVD signals must indicate that a valid voltage has been reached before the voltage indication is sent to the Flash.

54.10 Voltage monitor BIST programming

The application has to program the BIST_TIMExx registers according to the following timings, respect to the configured PBRIDGE_CLK:

- BIST_TIME10[Time_0] = 40 µs.
- BIST_TIME10[Time_1] = 50 µs.
- BIST_TIME32[Time_2] = 50 µs.
- BIST_TIME32[Time_3] = 50 µs.
- BIST_TIME65[Time_5] = 200 ns.
- BIST_TIME65[Time_6] = 200 ns.

In case that aforementioned timings are not applied, some voltage monitor BIST could trigger fake faults.

The user can run a BIST mode by setting the BIST_CTRL[START] register bit. The following voltage detectors are then serially tested:

- LVD100_SB
- MVD094_C
- MVD094_FA
- MVD094_FB
- LVD100_C
- LVD100_FL
- MVD270_C_MAIN
- MVD270_C_STBY
- MVD270_FL
- LVD290_C
- LVD290_FL
- LVD290_AS
- LVD290_IF
- LVD400_AS
- LVD400_IM
- LVD400_IF
- HVD134_C
- UVD140_C
- UVD140_FL
- HVD400_IF
- UVD600_FL
- UVD600_IF

Interrupt can be enabled by programming BIST_CTRL[IRQEN] and interrupt status is checked with BIST_CTRL[IRQST] on completion of BIST sequence.

FCCU event can be enabled by programming BIST_CTRL[NCFEN]. The BIST_CTRL[NCFST] status indicates BIST Fail on completion of BIST sequence.

BIST_DEBUG[VD_MON] register bits allow to know which voltage detector is under test.

The overall status of BIST is provided by the BIST_CTRL[STATUS] register bits.

The status of individual monitors is checked in flag registers BIST_FLAGS_PHASE1 (pass is all zero) and BIST_FLAGS_PHASE2 (pass is all one).

Note: When the SOC supply is 3.3V, the LVD400_xx monitors are not lifted but when we run BIST all the monitors are masked (during this time the event is seen in PMC_DIG_EPR_HV1 register). When the SOC supply is 5V, the HVD400_xx monitors are not lifted but when we run BIST, the event is seen in PMC_DIG_EPR_HV1 register.

55 Power Control Unit (MC_PCU)

55.1 Introduction

55.1.1 Overview

The power control unit (MC_PCU) is used to control the overall microcontroller power consumption. There are 4 power domains controlled by the MC_PCU which physically disconnects power to the domains when the specific operating mode allows this.

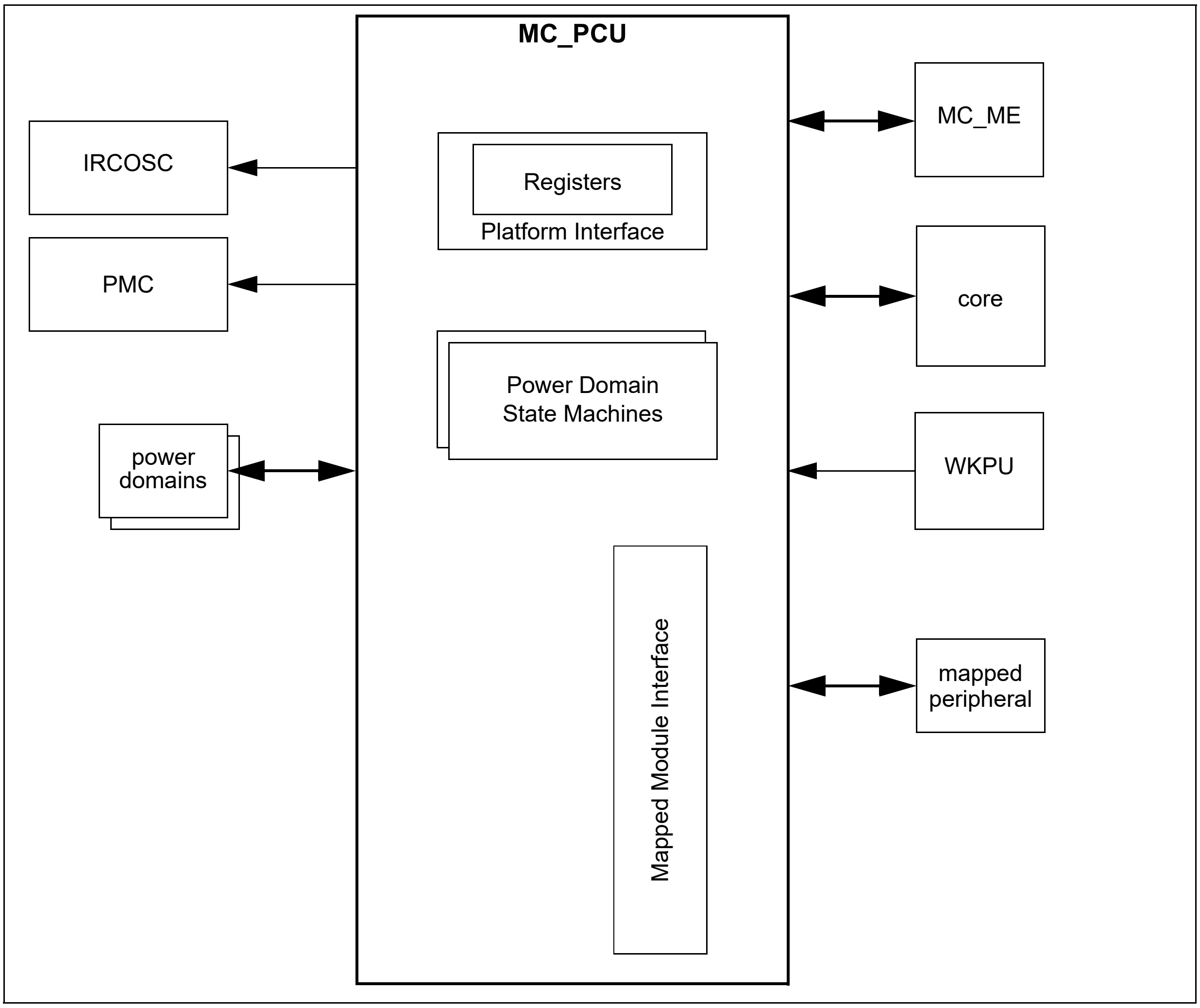
When a power domain is disconnected from the supply, the power consumption is reduced to zero in that domain. Any status information of such a power domain is lost. When re-connecting a power domain to the supply voltage, the domain draws an increased current until the power domain reaches its operational voltage.

For each mode, software can configure whether a power domain is connected to the supply voltage (power-up state) or disconnected (power-down state). Maximum power saving is reached by entering the STANDBY mode.

On each mode change request, the MC_PCU evaluates the power domain settings in the power domain configuration registers and initiates a power-down or a power-up sequence for each individual power domain. The power-up/down sequences are handled by finite state machines to ensure a smooth and safe transition from one power state to the other.

Figure 1138 depicts the MC_PCU block diagram.

Figure 1138. MC_PCU block diagram



55.1.2 Features

The MC_PCU includes:

- Support for 4 power domains
- Support for chip modes RESET, DRUN, SAFE, TEST, RUN0…3, HALT, STOP and STANDBY (for further mode details, refer to the MC_ME chapter)
- Power states updating on each mode change and on system wakeup
- A handshake mechanism for power state changes thus guaranteeing operable voltage

55.2 External Signal Description

The MC_PCU has no connections to any external pins.

55.3 Memory Map and Register Definition

55.3.1 Memory Map

Table 1166. MC_PCU Register Description

Address offset	Register name	Location
0x0000	Power Domain #0 Configuration (PCU_PCONF0)	*Section 55.3.2.1*
0x0004	Power Domain #1 Configuration (PCU_PCONF1)	*Section 55.3.2.2*
0x0008	Power Domain #2 Configuration (PCU_PCONF2)	*Section 55.3.2.3*
0x000C	Power Domain #3 Configuration (PCU_PCONF3)	*Section 55.3.2.4*
0x0010–0x003F	Reserved	
0x0040	Power Domain Status Register (PCU_PSTAT)	*Section 55.3.2.5*

Note: Any access to unused registers as well as write accesses to read-only registers will not change register content and will cause a transfer error

55.3.2 Register descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the PD0 field of the PCU_PSTAT register may be accessed as a word at address offset 0x0040, as a half-word at address offset 0x0042, or as a byte at address offset 0x0043.

55.3.2.1 Power Domain #0 Configuration Register (PCU_PCONF0)

Offset: 0x0000 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 1139. Power Domain #0 Configuration Register (PCU_PCONF0)

This register defines for power domain #0 whether it is on or off in each chip mode. As power domain #0 is the always-on power domain (and includes the MC_PCU), none of its bits are programmable. This register is available for completeness reasons.

Table 1167. PCU_PCONF0 register field descriptions

Field	Description
31 RST	Power domain control during RESET mode 0 Power domain off 1 Power domain on
30 TEST	Power domain control during TEST mode 0 Power domain off 1 Power domain on
29 SAFE	Power domain control during SAFE mode 0 Power domain off 1 Power domain on
28 DRUN	Power domain control during DRUN mode 0 Power domain off 1 Power domain on
27 RUN0	Power domain control during RUN0 mode 0 Power domain off 1 Power domain on
26 RUN1	Power domain control during RUN1 mode 0 Power domain off 1 Power domain on
25 RUN2	Power domain control during RUN2 mode 0 Power domain off 1 Power domain on
24 RUN3	Power domain control during RUN3 mode 0 Power domain off 1 Power domain on
23 HALT	Power domain control during HALT mode 0 Power domain off 1 Power domain on
21 STOP	Power domain control during STOP mode 0 Power domain off 1 Power domain on
18 STBY0	Power domain control during STANDBY mode 0 Power domain off 1 Power domain on

55.3.2.2 Power Domain #1 Configuration Register (PCU_PCONF1)

Offset: 0x0004 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 1140. Power Domain #1 Configuration Register (PCU_PCONF1)

This register defines for power domain #1 whether it is on or off in each chip mode. The bit field description is the same as in *Table 1167*. As the platform, clock generation, and mode control reside in power domain #1, this power domain is only powered down during the STANDBY mode. Therefore, none of the bits are programmable. This register is available for completeness reasons.

The difference between PCU_PCONF0 and PCU_PCONF1 is the reset value of the STBY0 bit: during the STANDBY mode, power domain #1 is disconnected from the power supply, and therefore PCU_PCONF1.STBY0 is always ‘0’. Power domain #0 is always on, and therefore PCU_PCONF0.STBY0 is ‘1’.

For further details about STANDBY mode, refer to *Section 55.4.4.1: STANDBY Mode transition*. NO_FEATURE

55.3.2.3 Power Domain #2 Configuration Register (PCU_PCONF2)

Offset: 0x0008 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W			STBY0			STOP		HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 1141. Power Domain #2 Configuration Register (PCU_PCONF2)

This register defines for power domain #2 whether it is on or off in each chip mode. The bit field description is the same as in *Table 1167*. This register controls the 120KB Standby RAM.

55.3.2.4 Power Domain #3 Configuration Register (PCU_PCONF3)

Offset: 0x000C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 1142. Power Domain #3 Configuration Register (PCU_PCONF3)

This register defines for power domain #3 whether it is on or off in each chip mode. The bit field description is the same as in *Table 1167*. This register controls the 96KB Standby RAM.

55.3.2.5 Power Domain Status Register (PCU_PSTAT)

Offset: 0x0040 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	1	PD3	PD2	PD1	PD0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 1143. Power Domain Status Register (PCU_PSTAT)

This register reflects the power status of all available power domains.

Table 1168. PCU_PSTAT field descriptions

Field	Description
31 PD0	Power status for power domain #0 0 Power domain is inoperable. 1 Power domain is operable.

55.4 Functional description

55.4.1 General

The MC_PCU controls all available power domains on a chip mode basis. The PCU_PCONF*n* registers specify during which system/user modes a power domain is powered up. The power state for each individual power domain is reflected by the bits in the PCU_PSTAT register.

On a mode change, the MC_PCU evaluates which power domain(s) must change power state. The power state is controlled by a state machine (FSM) for each individual power domain which ensures a clean and safe state transition.

55.4.2 Reset / Power-On Reset

After any reset, the chip will transition to the RESET mode during which all power domains are powered up (refer to the MC_ME chapter). Once the reset sequence has been completed, the DRUN mode is entered and software can begin the MC_PCU configuration.

55.4.3 MC_PCU configuration

By default, all power domains are powered in all modes other than STANDBY. Software can change the configuration for each power domain on a mode basis by programming the PCU_PCONF*n* registers.

Each power domain which is powered down is held in a reset state. Read/write accesses to peripherals in those power domains will result in a transfer error.

55.4.4 Mode transitions

On a mode change requested by the MC_ME, the MC_PCU evaluates the power configurations for all power domains. It compares the settings in the PCU_PCONF*n* registers for the new mode with the settings for the current mode. If the configuration for a power domain differs between the modes, a power state change request is generated. These requests are handled by a finite state machine to ensure a smooth and safe transition from one power state to another.

55.4.4.1 STANDBY Mode transition

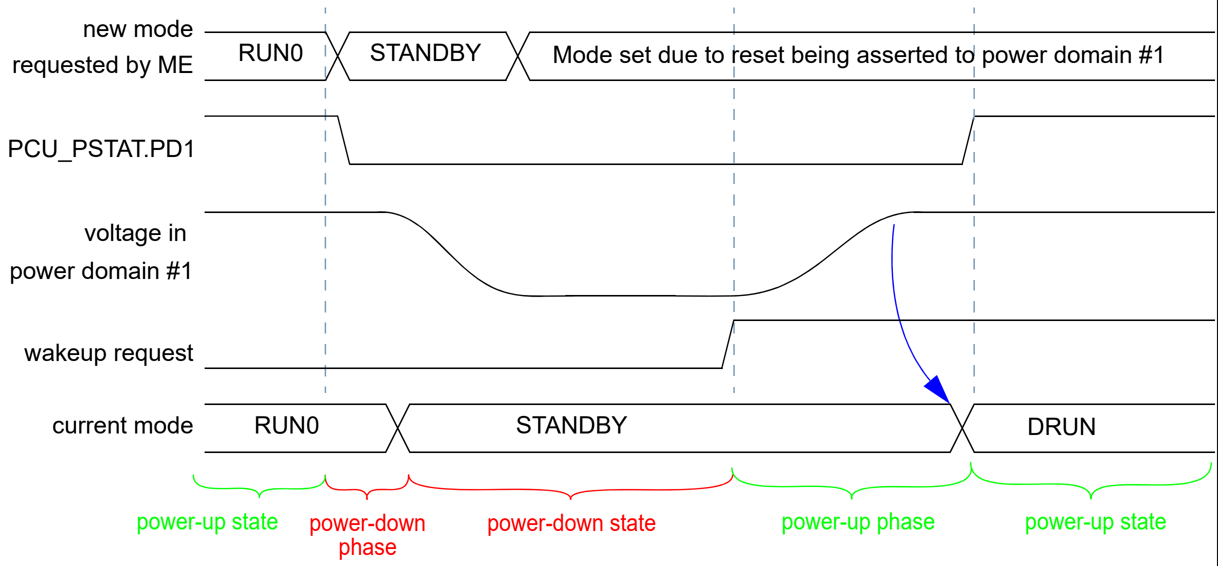
STANDBY offers the maximum power saving. The level of power saving is software-controllable via the settings in the PCU_PCONF*n* registers for power domain #2 onwards. Power domain #0 stays connected to the power supply while power domain #1 is disconnected from the power supply. Amongst others power domain #1 contains the platform and the MC_ME. Therefore this mode differs from all other user/system modes.

Once STANDBY is entered it can only be left via a system wakeup. On exiting the STANDBY mode, all power domains are powered up according to the settings in the PCU_PCONF*n* registers, and the DRUN mode is entered. In DRUN mode, both domains #0 and #1 are powered.

Figure 1144 shows an example for a mode transition from RUN0 to STANDBY to DRUN. All power domains which have PCU_PCONF*n*.STBY cleared will enter power-down phase. In this example power domain #1 will be disabled during STANDBY mode.

When the MC_PCU receives the mode change request to STANDBY mode it starts the power down phase for power domain #1. During the power down phase, clocks are disabled and reset is asserted resulting in a loss of all information for this power domain. Then the power domain is disconnected from the power supply (power-down state).

Figure 1144. MC_PCU Events During Power Sequences (STANDBY mode)



Notes:

Not drawn to scale; PCONF1.RUN0 = 1; PCONF1.STBY0 = 0

When the MC_PCU receives a system wakeup request, it starts the power-up phase. The power domain is re-connected to the power supply and the voltage in power domain #1 will increase slowly. Once the voltage is in an operable range, clocks are enabled and the reset is be deasserted (power-up state).

Prior to standby entry, PCTL for WKPU should be disabled.

Note: It is possible that due to a wakeup request, power-up is requested before a power domain completed its power-down sequence. In this case, the information in that power domain is lost.

Exiting the STANDBY mode can only be done via a system wakeup event from:

- API/SSWU module (mapped on WKUP0 line),
- RTC timer (mapped on WKUP1 line),
- external wakeup pads (WKUPx line, where x>1).

Refer to the wakeup vector mapping table in the Device Configuration chapter for complete wakeup lines list and to SPC584Bx IO_Definition document for pad/wakeup lines mapping.

55.4.4.2 Power Saving for Memories During STANDBY Mode

All memories which are not powered down during STANDBY mode automatically enter a power saving state. No software configuration is required to enable this power saving state. While a memory is residing in this state an increased power saving is achieved. Data in the memories is retained.

55.5 Application information

55.5.1 STANDBY mode considerations

STANDBY offers maximum power saving possibility. But power is only saved during the time a power domain is disconnected from the supply. Increased power is required when a power domain is re-connected to the power supply. Additional power is required during restoring the information (for example in the platform). Care should be taken that the time during which the chip is operating in STANDBY mode is significantly longer than the required time for restoring the information.

56 Mode Entry Module (MC_ME)

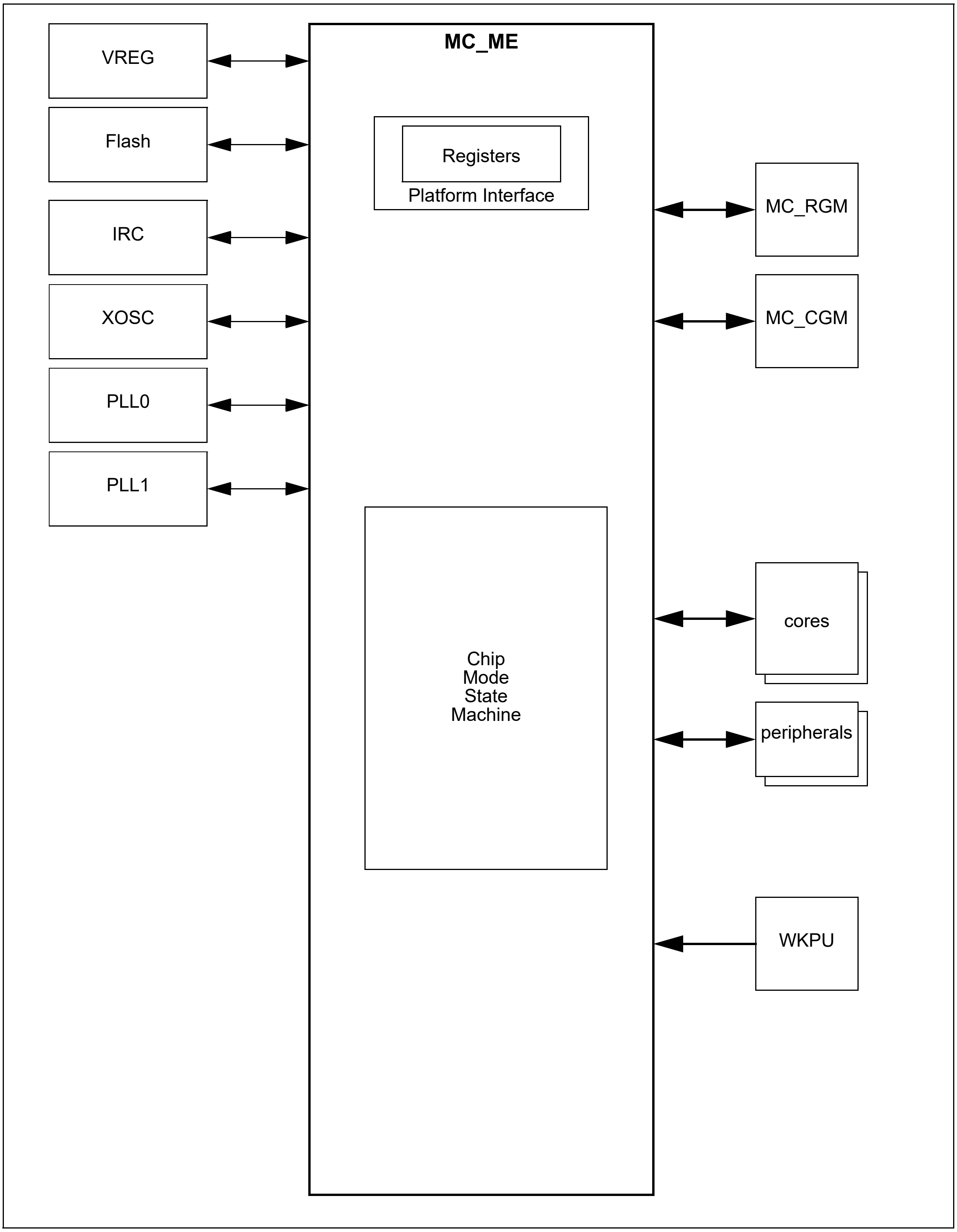
56.1 Introduction

56.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

Figure 1145 shows the MC_ME block diagram.

Figure 1145. MC_ME block diagram



56.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the ME_ME register
- definition of various chip mode configurations by the ME_<mode>_MC registers
- control of the actual chip mode by the ME_MCTL register
- capture of the current mode and various resource status within the contents of the ME_GS register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTL*n* registers
- additional core clock gating and boot address control based on the ME_CCTL*n* and ME_CADDR*n* registers
- capture of current peripheral and additional core clock gated/enabled status
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

56.1.3 Modes of operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as: RESET, DRUN, SAFE, and TEST. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as: RUN0–3, HALT0, STOP0, and STANDBY0 which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, SAFE, TEST, and RUN0–3 are the chip software running modes.

Table 1169 describes the MC_ME modes.

Table 1169. MC_ME mode descriptions

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and Flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAF, when present, is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from SAFE, TEST and RUN0–3, wakeup request from STANDBY0	system reset assertion, RUN0–3, TEST, STANDBY0 via software, SAFE via software or hardware failure.

Table 1169. MC_ME mode descriptions (continued)

Name	Description	Entry	Exit
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN, TEST, and RUN0–3	system reset assertion, DRUN via software
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from DRUN	system reset assertion, DRUN via software
RUN0–3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from DRUN or other RUN0–3, interrupt event from HALT0, interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, other RUN0–3 modes, HALT0, STOP0, STANDBY0 via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals such as: clock sources, Flash, main regulator, for efficient power management at the cost of higher wakeup latency.	software request from RUN0–3	system reset assertion, SAFE on hardware failure, RUN0–3 on interrupt event
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0–3	system reset assertion, SAFE on hardware failure, RUN0–3 on interrupt event or wakeup event
STANDBY0	This is a reduced-leakage low-power mode during which power supply is cut off from most of the chip. Wakeup from this mode takes a relatively long time, and content is lost or must be restored from backup.	software request from RUN0–3, DRUN modes	system reset assertion, DRUN on wakeup event

56.2 External signal description

The MC_ME has no connections to any external pins.

56.3 Memory map and register definition

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of additional core enabling and disabling control and status reporting

56.3.1 Memory map

Table 1170. MC_ME register description

Address offset	Name	Description	Location
0x000	ME_GS	Global Status	Section 56.3.2.1
0x004	ME_MCTL	Mode Control	Section 56.3.2.2
0x008	ME_ME	Mode Enable	Section 56.3.2.3
0x00C	ME_IS	Interrupt Status	Section 56.3.2.4
0x010	ME_IM	Interrupt Mask	Section 56.3.2.5
0x014	ME_IMTS	Invalid Mode Transition Status	Section 56.3.2.6
0x018	ME_DMTS	Debug Mode Transition Status	Section 56.3.2.7
0x01C–0x1F	Reserved		
0x020	ME_RESET_MC	RESET Mode Configuration	Section 56.3.2.8
0x024	ME_TEST_MC	TEST Mode Configuration	Section 56.3.2.9
0x028	ME_SAFE_MC	SAFE Mode Configuration	Section 56.3.2.10
0x02C	ME_DRUN_MC	DRUN Mode Configuration	Section 56.3.2.11
0x030	ME_RUN0_MC	RUN0 Mode Configuration	Section 56.3.2.12
0x034	ME_RUN1_MC	RUN1 Mode Configuration	Section 56.3.2.12
0x038	ME_RUN2_MC	RUN2 Mode Configuration	Section 56.3.2.12
0x03C	ME_RUN3_MC	RUN3 Mode Configuration	Section 56.3.2.12
0x040	ME_HALT0_MC	HALT0 Mode Configuration	Section 56.3.2.13
0x044–0x47	Reserved		
0x048	ME_STOP0_MC	STOP0 Mode Configuration	Section 56.3.2.14
0x04C–0x53	Reserved		
0x054	ME_STANDBY0_MC	STANDBY0 Mode Configuration	Section 56.3.2.15
0x058–0x5F	Reserved		
0x060	ME_PS0	Peripheral Status 0	Section 56.3.2.16
0x064	ME_PS1	Peripheral Status 1	Section 56.3.2.17
0x068	ME_PS2	Peripheral Status 2	Section 56.3.2.18
0x06C	ME_PS3	Peripheral Status 3	Section 56.3.2.19
0x070	ME_PS4	Peripheral Status 4	Section 56.3.2.20
0x074	ME_PS5	Peripheral Status 5	Section 56.3.2.21
0x078	ME_PS6	Peripheral Status 6	Section 56.3.2.22
0x07C	ME_PS7	Peripheral Status 7	Section 56.3.2.23
0x080	ME_RUN_PC0	Run Peripheral Configuration 0	Section 56.3.2.24
0x084	ME_RUN_PC1	Run Peripheral Configuration 1	Section 56.3.2.24
0x088	ME_RUN_PC2	Run Peripheral Configuration 2	Section 56.3.2.24

Table 1170. MC_ME register description (continued)

Address offset	Name	Description	Location
0x08C	ME_RUN_PC3	Run Peripheral Configuration 3	Section 56.3.2.24
0x090	ME_RUN_PC4	Run Peripheral Configuration 4	Section 56.3.2.24
0x094	ME_RUN_PC5	Run Peripheral Configuration 5	Section 56.3.2.24
0x098	ME_RUN_PC6	Run Peripheral Configuration 6	Section 56.3.2.24
0x09C	ME_RUN_PC7	Run Peripheral Configuration 7	Section 56.3.2.24
0x0A0	ME_LP_PC0	Low-Power Peripheral Configuration 0	Section 56.3.2.25
0x0A4	ME_LP_PC1	Low-Power Peripheral Configuration 1	Section 56.3.2.25
0x0A8	ME_LP_PC2	Low-Power Peripheral Configuration 2	Section 56.3.2.25
0x0AC	ME_LP_PC3	Low-Power Peripheral Configuration 3	Section 56.3.2.25
0x0B0	ME_LP_PC4	Low-Power Peripheral Configuration 4	Section 56.3.2.25
0x0B4	ME_LP_PC5	Low-Power Peripheral Configuration 5	Section 56.3.2.25
0x0B8	ME_LP_PC6	Low-Power Peripheral Configuration 6	Section 56.3.2.25
0x0BC	ME_LP_PC7	Low-Power Peripheral Configuration 7	Section 56.3.2.25
0x0C0–0x0CE	ME_PCTL0–ME_PCTL14	Reserved	
0x0CF	ME_PCTL15	S_SIUL2 Control	Section 56.3.2.26
0x0D0–0x0DD	ME_PCTL16–ME_PCTL29	Reserved	
0x0DE	ME_PCTL30	S_PIT_0 Control	Section 56.3.2.26
0x0DF–0x0E3	ME_PCTL31–ME_PCTL35	Reserved	
0x0E4	ME_PCTL36	S_DMAMUX_0 Control	Section 56.3.2.26
0x0E5	ME_PCTL37	Reserved	
0x0E6	ME_PCTL38	S_CRC_0 Control	Section 56.3.2.26
0x0E7–0x0FD	ME_PCTL39–ME_PCTL61	Reserved	
0x0FE	ME_PCTL62	S_CCCU Control	Section 56.3.2.26
0x0FF–0x102	ME_PCTL63–ME_PCTL66	Reserved	
0x103	ME_PCTL67	S_CAN_SUB_0_MCAN_3 Control	Section 56.3.2.26
0x104	ME_PCTL68	S_CAN_SUB_0_MCAN_2 Control	Section 56.3.2.26
0x105	ME_PCTL69	S_CAN_SUB_0_MCAN_1 Control	Section 56.3.2.26
0x106–0x107	ME_PCTL70–ME_PCTL71	Reserved	
0x108	ME_PCTL72	S_CAN_SUB_0_MCAN_0 Control	Section 56.3.2.26
0x109	ME_PCTL73	Reserved	
0x10A	ME_PCTL74	S_CAN_SUB_0_MESSAGE_RAM Control	Section 56.3.2.26
0x10B–0x115	ME_PCTL75–ME_PCTL85	Reserved	
0x116	ME_PCTL86	S_LINFlexD_12 Control	Section 56.3.2.26
0x117	ME_PCTL87	S_LINFlexD_10 Control	Section 56.3.2.26
0x118	ME_PCTL88	S_LINFlexD_8 Control	Section 56.3.2.26

Table 1170. MC_ME register description (continued)

Address offset	Name	Description	Location
0x119	ME_PCTL89	S_LINFlexD_6 Control	Section 56.3.2.26
0x11A	ME_PCTL90	S_LINFlexD_4 Control	Section 56.3.2.26
0x11B	ME_PCTL91	S_LINFlexD_2 Control	Section 56.3.2.26
0x11C	ME_PCTL92	S_LINFlexD_0 Control	Section 56.3.2.26
0x11D–0x11F	ME_PCTL93–ME_PCTL95	Reserved	
0x120	ME_PCTL96	S_DSPI_6 Control	Section 56.3.2.26
0x121	ME_PCTL97	S_DSPI_4 Control	Section 56.3.2.26
0x122	ME_PCTL98	S_DSPI_2 Control	Section 56.3.2.26
0x123	ME_PCTL99	S_DSPI_0 Control	Section 56.3.2.26
0x124	ME_PCTL100	Reserved	
0x125	ME_PCTL101	S_IIC_0 Control	Section 56.3.2.26
0x126–0x12F	ME_PCTL102–ME_PCTL111	Reserved	
0x130	ME_PCTL112	S_SAR_ADC_B Control (SAR_ADC_12bit_SUPERVISOR)	Section 56.3.2.26
0x131	ME_PCTL113	Reserved	
0x132	ME_PCTL114	S_SAR_ADC_10bit_STDBY Control	Section 56.3.2.26
0x133–0x13E	ME_PCTL115–ME_PCTL126	Reserved	
0x13F	ME_PCTL127	S_SAR_ADC_12bit_0 Control	Section 56.3.2.26
0x140	ME_PCTL128	Reserved	
0x141	ME_PCTL129	S_SAR_ADC_B seq Control (SAR_ADC_12bit_SUPERVISOR_seq)	Section 56.3.2.26
0x142	ME_PCTL130	S_SAR_ADC_10bit_STDBY_seq Control	Section 56.3.2.26
0x143–0x14C	ME_PCTL131–ME_PCTL140	Reserved	
0x14D	ME_PCTL141	S_SAR_ADC_12bit_1_seq Control	Section 56.3.2.26
0x14E	ME_PCTL142	Reserved	
0x14F	ME_PCTL143	S_SAR_ADC_12bit_0_seq Control	Section 56.3.2.26
0x150–0x15D	ME_PCTL144–ME_PCTL157	Reserved	
0x15E	ME_PCTL158	S_PIT_1 Control	Section 56.3.2.26
0x15F–0x163	ME_PCTL159–ME_PCTL163	Reserved	
0x164	ME_PCTL164	S_DMAMUX_1 Control	Section 56.3.2.26
0x165	ME_PCTL165	Reserved	
0x166	ME_PCTL166	S_CRC_1 Control	Section 56.3.2.26
0x167–0x182	ME_PCTL167–ME_PCTL194	Reserved	
0x183	ME_PCTL195	S_CAN_SUB_1_MCAN_4 Control	Section 56.3.2.26
0x184	ME_PCTL196	S_CAN_SUB_1_MCAN_3 Control	Section 56.3.2.26

Table 1170. MC_ME register description (continued)

Address offset	Name	Description	Location
0x185	ME_PCTL197	S_CAN_SUB_1_MCAN_2 Control	Section 56.3.2.26
0x186	ME_PCTL198	S_CAN_SUB_1_MCAN_1 Control	Section 56.3.2.26
0x187–0x189	ME_PCTL199–ME_PCTL201	Reserved	
0x18A	ME_PCTL202	S_CAN_SUB_1_MESSAGE_RAM Control	Section 56.3.2.26
0x18B–0x194	ME_PCTL203–ME_PCTL212	Reserved	
0x195	ME_PCTL213	S_LINFlexD_15 Control	Section 56.3.2.26
0x196	ME_PCTL214	Reserved	
0x197	ME_PCTL215	S_LINFlexD_11 Control	Section 56.3.2.26
0x198	ME_PCTL216	S_LINFlexD_9 Control	Section 56.3.2.26
0x199	ME_PCTL217	S_LINFlexD_7 Control	Section 56.3.2.26
0x19A	ME_PCTL218	S_LINFlexD_5 Control	Section 56.3.2.26
0x19B	ME_PCTL219	S_LINFlexD_3 Control	Section 56.3.2.26
0x19C	ME_PCTL220	S_LINFlexD_1 Control	Section 56.3.2.26
0x19D–0x1A0	ME_PCTL221–ME_PCTL224	Reserved	
0x1A1	ME_PCTL225	S_DSPI_5 Control	Section 56.3.2.26
0x1A2	ME_PCTL226	S_DSPI_3 Control	Section 56.3.2.26
0x1A3	ME_PCTL227	S_DSPI_1 Control	Section 56.3.2.26
0x1A4–0x1A9	ME_PCTL228–ME_PCTL233	Reserved	
0x1AA	ME_PCTL234	S_ETHERNET_0 Control	Section 56.3.2.26
0x1AB–0x1AF	ME_PCTL235–ME_PCTL239	Reserved	
0x1B0	ME_PCTL240	S_eMIOS_1 Control	Section 56.3.2.26
0x1B1	ME_PCTL241	S_eMIOS_0 Control	Section 56.3.2.26
0x1B2	ME_PCTL242	S_STDBY_CTU_0 Control	Section 56.3.2.26
0x1B3	ME_PCTL243	S_BODY_CTU_0 Control	Section 56.3.2.26
0x1B4–0x1BE	ME_PCTL244–ME_PCTL254	Reserved	
0x1BF	ME_PCTL255	S_SAR_ADC_12bit_1 Control	Section 56.3.2.26
0x1C0	ME_CS	Core Status	Section 56.3.2.27
0x1C4	ME_CCTL0	CORE0 Control	Section 56.3.2.28
0x1C6–0x1CB	Reserved		
0x1CC	ME_CCTL4	CORE4 Control	Section 56.3.2.29
0x1CE–0x1DF	Reserved		
0x1E0	ME_CADDR0	CORE0 Address	Section 56.3.2.30
0x1E4–0x1EF	Reserved		
0x1F0	ME_CADDR4	CORE4 Address	Section 56.3.2.31

Note: Any access to unused registers as well as write accesses to read-only registers will:

- *not change register content*
- *cause a transfer error*

56.3.2 Register description

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Some fields may be read-only, and their reset value of ‘1’ or ‘0’ and the corresponding behavior cannot be changed.

56.3.2.1 Global Status Register (ME_GS)

This register contains global mode status.

Offset : 0x000 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_CURRENT_MODE				S_MTRANS	1	0	0	S_PDO	0	0	S_MVR	0	0	S_FLA	
W																
Reset	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	S_PLL1	S_PLL0	S_XOSC	S_IRC	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1146. Global Status Register (ME_GS)

Table 1171. ME_GS field descriptions

Field	Description
0:3 S_CURRENT_MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 STANDBY0 1110 reserved 1111 reserved
4 S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
8 S_PDO	Output power-down status This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In STOP0 mode, the slew rate control mechanism of the output pads is disabled to reduce power consumption, but the state of the output remains functional.
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
14:15 S_FLA	Flash availability status 00 Flash is not available 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode and available for use
24 S_PLL1	secondary PLL status 0 secondary PLL is not stable 1 secondary PLL is providing a stable clock
25 S_PLL0	primary PLL status 0 primary PLL is not stable 1 primary PLL is providing a stable clock
26 S_XOSC	external crystal oscillator status 0 external crystal oscillator is not stable 1 external crystal oscillator is providing a stable clock

Table 1171. ME_GS field descriptions (continued)

Field	Description
27 S_IRC	16 MHz internal RC oscillator status 0 16 MHz internal RC oscillator is not stable 1 16 MHz internal RC oscillator is providing a stable clock
28:31 S_SYSCLK	System clock switch status These bits specify the system clock currently used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

56.3.2.2 Mode Control Register (ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<mode>_MC registers must respect this for successful mode requests.

Note: Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

Offset: 0x004 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TARGET_MODE				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
W	KEY															
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Figure 1147. Mode Control Register (ME_MCTL)

Table 1172. ME_MCTL field descriptions

Field	Description
0:3 TARGET_MODE	Target chip mode These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0–3 mode value. 0000 RESET (triggers a ‘functional’ reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 STANDBY0 1110 reserved 1111 RESET (triggers a ‘destructive’ reset event)
16:31 KEY	Control key These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101 1010 1111 0000 (0x5AF0) INVERTED KEY: 1010 0101 0000 1111 (0xA50F)

56.3.2.3 Mode Enable Register (ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. RESET, SAFE, DRUN, and RUN0 modes are always enabled.

Offset: 0x008 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESET_DEST	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET_FUNC
W																
Reset	1	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1

Figure 1148. Mode Enable Register (ME_ME)

Table 1173. ME_ME Field Descriptions

Field	Description
16 RESET_DEST	‘destructive’ RESET mode enable 1 ‘destructive’ RESET mode is enabled
18 STANDBY0	STANDBY0 mode enable 0 STANDBY0 mode is disabled 1 STANDBY0 mode is enabled
21 STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
23 HALT0	HALT0 mode enable 0 HALT0 mode is disabled 1 HALT0 mode is enabled
24 RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
25 RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
26 RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
27 RUN0	RUN0 mode enable 1 RUN0 mode is enabled
28 DRUN	DRUN mode enable 1 DRUN mode is enabled
29 SAFE	SAFE mode enable 1 SAFE mode is enabled

Table 1173. ME_ME Field Descriptions (continued)

Field	Description
30 TEST	TEST mode enable 0 TEST mode is disabled 1 TEST mode is enabled
31 RESET_FUNC	'functional' RESET mode enable 1 'functional' RESET mode is enabled

56.3.2.4 Interrupt Status Register (ME_IS)

This register provides the current interrupt status.

Offset: 0x00C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	I_ICONF_CC	I_ICONF_CU	I_ICONF	I_IMODE	I_SAFE	I_MTC
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1149. Interrupt Status Register (ME_IS)

Table 1174. ME_IS field descriptions

Field	Description
26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) This bit is set if a write access to one of the ME_CCTLn registers is attempted while a mode transition is in progress. 0 No write to an ME_CADDRn register was attempted during an ongoing mode transition 1 A write to an ME_CADDRn register was attempted during an ongoing mode transition
27 I_ICONF_CU	Invalid mode configuration interrupt (Clock Usage) This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration (clock usage) interrupt occurred 1 Invalid mode configuration (clock usage) interrupt is pending

Table 1174. ME_IS field descriptions (continued)

Field	Description
28 I_ICONF	Invalid mode configuration interrupt This bit is set whenever a write operation to ME_<*mode*>_MC registers with invalid mode configuration is attempted. It is cleared by writing a ‘1’ to this bit. 0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending
29 I_IMODE	Invalid mode interrupt This bit is set whenever an invalid mode transition is requested. It is cleared by writing a ‘1’ to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
30 I_SAFE	SAFE mode interrupt This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a ‘1’ to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
31 I_MTC	Mode transition complete interrupt This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a ‘1’ to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, STOP0, or STANDBY0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

56.3.2.5 Interrupt Mask Register (ME_IM)

This register controls whether an event generates an interrupt or not.

Offset: 0x010 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	M_ICONF_CC	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1150. Interrupt Mask Register (ME_IM)

Table 1175. ME_IM field descriptions

Field	Description
26 M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
27 M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
28 M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
30 M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
31 M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

56.3.2.6 Invalid Mode Transition Status Register (ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Offset: 0x014 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	S_MRIG	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1151. Invalid Mode Transition Status Register (ME_IMTS)

Table 1176. ME_IMTS field descriptions

Field	Description
26 S_MRIG	Mode Request Ignored status This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress, a transition to STOP0 mode is requested while a wakeup or interrupt request is active, a transition to HALT0 mode is requested while an interrupt request is active, or a transition to STANDBY mode is requested while a wakeup request is active. It is cleared by writing a ‘1’ to this bit. 0 Mode transition requested is not ignored 1 Mode transition requested is ignored
27 S_MTI	Mode Transition Illegal status This bit is set whenever a new mode is requested while some other mode transition process is active (S_MTRANS is ‘1’). Please see *Section 56.4.5: Mode Transition Interrupts*, for the exceptions to this behavior. It is cleared by writing a ‘1’ to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
28 S_MRI	Mode Request Illegal status This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
29 S_DMA	Disabled Mode Access status This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
30 S_NMA	**Non-existing Mode Access status** This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
31 S_SEA	SAFE Event Active status This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a ‘1’ to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

56.3.2.7 Debug Mode Transition Status Register (ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS.S_MTRANS may be taking longer than expected.

Note: The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Offset: 0x018 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0	0	0	0	MPH_BUSY	0	0	PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDP_PRPH_0_255	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	Reserved	FLASH_SC	CDP_PRPH_224_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1152. Debug Mode Transition Status Register (ME_DMTS)

Table 1177. ME_DMTS field descriptions

Field	Description
0:3 PREVIOUS_MODE	Previous chip mode These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 STANDBY0 1110 reserved 1111 reserved
8 MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its power-up/down sequencing. It is cleared when the MC_PCU has responded. 0 Handshake is not busy 1 Handshake is busy

Table 1177. ME_DMTS field descriptions (continued)

Field	Description
11 PMC_PROG	MC_PCU Mode Change in Progress indicator This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
12 DBG_MODE	Debug mode indicator This bit is set while the chip is in debug mode. 0 The chip is not in debug mode 1 The chip is in debug mode
13 CCKL_PROG	Core Clock Enable/Disable in Progress This bit is set while any core's clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled 1 A core clock is being enabled or disabled
14 PCS_PROG	Progressive System Clock Switching in Progress This bit is set while the progressive system clock switching process is in progress. 0 PCS is not in progress 1 PCS is in progress
15 SMR	SAFE mode request from MC_RGM is active indicator This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared. 0 A SAFE mode request is not active 1 A SAFE mode request is active
16 CDP_PRPH_0_255	Clock Disable Process Pending status for Peripherals 0–255(1) This bit is set when any peripheral appearing has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
17 VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
18 CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place

Table 1177. ME_DMTS field descriptions (continued)

Field	Description
19 IRC_SC	IRC State Change during mode transition indicator This bit is set when the 16 MHz internal RC oscillator is requested to change its power up/down state. It is cleared when the 16 MHz internal RC oscillator has completed its state change. 0 No state change is taking place 1 A state change is taking place
20 SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A secondary clock source is a clock source other than IRC.) 0 No state change is taking place 1 A state change is taking place
21 SYSCLK_SW	System Clock Switching pending status 0 No system clock source switching is pending 1 A system clock source switching is pending
23 FLASH_SC	FLASH State Change during mode transition indicator This bit is set when the FLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
24 CDP_PRPH_224_255	Clock Disable Process Pending status for Peripherals 224–255(1) This bit is set when any peripheral appearing in ME_PS7 has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
25 CDP_PRPH_192_223	Clock Disable Process Pending status for Peripherals 192–223 This bit is set when any peripheral appearing in ME_PS6 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
26 CDP_PRPH_160_191	Clock Disable Process Pending status for Peripherals 160–191 This bit is set when any peripheral appearing in ME_PS5 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table 1177. ME_DMTS field descriptions (continued)

Field	Description
27 CDP_PRPH_128_159	Clock Disable Process Pending status for Peripherals 128–159 This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
28 CDP_PRPH_96_127	Clock Disable Process Pending status for Peripherals 96–127 This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
29 CDP_PRPH_64_95	Clock Disable Process Pending status for Peripherals 64–95 This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
30 CDP_PRPH_32_63	Clock Disable Process Pending status for Peripherals 32–63 This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
31 CDP_PRPH_0_31	Clock Disable Process Pending status for Peripherals 0–31 This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

1. Peripheral n corresponds to the ME_PCTLn register. Please refer to *Table 1170* for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register.

56.3.2.8 RESET Mode Configuration Register (ME_RESET_MC)

This register configures system behavior during RESET mode. See *Table 1178* for details.

Note: The following configuration values are set according to the chip configuration:

XOSCON

Offset: 0x020 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1153. RESET Mode Configuration Register (ME_RESET_MC)

56.3.2.9 TEST Mode Configuration Register (ME_TEST_MC)

This register configures system behavior during TEST mode. See *Table 1178* for details.

Note: Byte write accesses are not allowed to this register.

Offset: 0x024 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL(1)			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Only bit [3] have an effect on progressive clock switching and bits [1:2] are ignored.

Figure 1154. TEST Mode Configuration Register (ME_TEST_MC)

56.3.2.10 SAFE Mode Configuration Register (ME_SAFE_MC)

This register configures system behavior during SAFE mode. See *Table 1178* for details.

Note: Byte write accesses are not allowed to this register.

Offset: 0x028 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL(1)			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Only bit [3] have an effect on progressive clock switching and bits [1:2] are ignored.

Figure 1155. SAFE Mode Configuration Register (ME_SAFE_MC)

56.3.2.11 DRUN Mode Configuration Register (ME_DRUN_MC)

This register configures system behavior during DRUN mode. See Table 1178 for details.

Note: Byte write accesses are not allowed to this register.

The following configuration values are set according to the chip configuration:

XOSCON

Offset: 0x02C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL(1)			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Only bit [3] have an effect on progressive clock switching and bits [1:2] are ignored.

Figure 1156. DRUN Mode Configuration Register (ME_DRUN_MC)

56.3.2.12 RUN*n* Mode Configuration Registers (ME_RUN*n*_MC)

This register configures system behavior during RUN0–3 modes. See Table 1178 for details.

Note: Byte write accesses are not allowed to this register.

Offset: 0x030 + *n**0x4 (*n* = 0 to 3) Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL(1)			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Only bit [3] have an effect on progressive clock switching and bits [1:2] are ignored.

Figure 1157. RUN0–3 Mode Configuration Registers (ME_RUN*n*_MC)

56.3.2.13 HALT0 Mode Configuration Register (ME_HALT0_MC)

This register configures system behavior during HALT0 mode. See Table 1178 for details.

Note: Byte write accesses are not allowed to this register.

Offset: 0x040 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1158. HALT0 Mode Configuration Register (ME_HALT0_MC)

56.3.2.14 STOP0 Mode Configuration Register (ME_STOP0_MC)

This register configures system behavior during STOP0 mode. See Table 1178 for details.

Note: Byte write accesses are not allowed to this register.

Offset: 0x048 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1159. STOP0 Mode Configuration Register (ME_STOP0_MC)

56.3.2.15 STANDBY0 Mode Configuration Register (ME_STANDBY0_MC)

Note: This register configures system behavior during STANDBY0 mode. Please refer to *Table 1178* for details.*Byte write accesses are not allowed to this register.*

Offset: 0x054 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 1160. STANDBY0 Mode Configuration Register (ME_STANDBY0_MC)

Table 1178. ME_<*mode*>_MC Field Descriptions

Field	Description
1:3 PWRLVL	Power level This bit indicates the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled. **Note:** For ME_RESET_MC, ME_HALT0_MC, ME_STOP0_MC and ME_STANDBY0_MC registers this bit field is read-only and always 0.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, internal weak pull-ups of outputs are enabled and pads power sequence driver is disabled The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional. In STANDBY0 mode, power sequence driver and all pads except those mapped on wakeup lines are not powered and therefore high impedance. Wakeup line configuration remains unchanged
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the Flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	external crystal oscillator control 0 external crystal oscillator is switched off 1 external crystal oscillator is switched on

Table 1178. ME_<*mode*>_MC Field Descriptions (continued)

Field	Description
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

56.3.2.16 Peripheral Status Register 0 (ME_PS0)

This register provides the status of the peripherals. See *Table 1179* for details.

Offset: 0x060 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_PIT_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_SIUL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1161. Peripheral Status Register 0 (ME_PS0)

56.3.2.17 Peripheral Status Register 1 (ME_PS1)

This register provides the status of the peripherals. See *Table 1179* for details.

Offset: 0x064 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_CCCU	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	S_CRC_0	0	S_DMAMUX_0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1162. Peripheral Status Register 1 (ME_PS1)

56.3.2.18 Peripheral Status Register 2 (ME_PS2)

This register provides the status of the peripherals. See *Table 1179* for details.

Offset: 0x068 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_LINFlexD_0	S_LINFlexD_2	S_LINFlexD_4	S_LINFlexD_6	S_LINFlexD_8	S_LINFlexD_10	S_LINFlexD_12	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_CANSUBSYS0_CAN_RAM_CTR	0	S_CANSUBSYS0_MCAN_0	0	0	S_CANSUBSYS0_MCAN_1	S_CANSUBSYS0_MCAN_2	S_CANSUBSYS0_MCAN_3	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1163. Peripheral Status Register 2 (ME_PS2)

56.3.2.19 Peripheral Status Register 3 (ME_PS3)

This register provides the status of the peripherals. See *Table 1179* for details.

Address 0x06C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_SAR_ADC_12BIT_0	0	0	0	0	0	0	0	0	0	0	0	0	S_SAR_ADC_STDBY	0	S_SAR_ADC_B
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	S_IIC_0	0	S_DSPI_0	S_DSPI_2	S_DSPI_4	S_DSPI_6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1164. Peripheral Status Register 3 (ME_PS3)

56.3.2.20 Peripheral Status Register 4 (ME_PS4)

This register provides the status of the peripherals. See *Table 1179* for details.

Offset: 0x070 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_PIT_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_SAR_ADC_12BIT_SEQ_0	0	S_SAR_ADC_12BIT_SEQ_1	0	0	0	0	0	0	0	0	0	0	S_SAR_ADC_SEQ_STDBY	S_SAR_ADC_SEQ_B	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1165. Peripheral Status Register 4 (ME_PS4)

56.3.2.21 Peripheral Status Register 5 (ME_PS5)

This register provides the status of the peripherals. See *Table 1179* for details.

Offset: 0x074 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	S_CRC_1	0	S_DMAMUX_1	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1166. Peripheral Status Register 5 (ME_PS5)

56.3.2.22 Peripheral Status Register 6 (ME_PS6)

This register provides the status of the peripherals. See *Table 1179* for details.

Offset: 0x078 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_LINFlexD_1	S_LINFlexD_3	S_LINFlexD_5	S_LINFlexD_7	S_LINFlexD_9	S_LINFlexD_11	0	S_LINFlexD_15	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_CANSUBSYS1_CAN_RAM_CTR	0	0	0	S_CANSUBSYS1_MCAN_1	S_CANSUBSYS1_MCAN_2	S_CANSUBSYS1_MCAN_3	S_CANSUBSYS1_MCAN_4	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1167. Peripheral Status Register 6 (ME_PS6)

56.3.2.23 Peripheral Status Register 7 (ME_PS7)

This register provides the status of the peripherals. See *Table 1179* for details.

Offset: 0x07C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_SAR_ADC_12BIT_1	0	0	0	0	0	0	0	0	0	0	0	S_BODY_CTU_0	S_STDBY_CTU_0	S_EMIOS_0	S_EMIOS_1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_ETHERNET_0	0	0	0	0	0	0	S_DSPI_1	S_DSPI_3	S_DSPI_5	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1168. Peripheral Status Register 7 (ME_PS7)

Table 1179. ME_PS*n* field descriptions

Field	Description
S_*<periph>*	Peripheral status These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

56.3.2.24 Run Peripheral Configuration Registers (ME_RUN_PC*n*)

These registers configure eight different types of peripheral behavior during run modes.

Offset: 0x080 + *n**0x4 (n = 0 to 7) Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1169. Run Peripheral Configuration Registers (ME_RUN_PC*n*)

Table 1180. ME_RUN_PC*n* field descriptions

Field	Description
24 RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
25 RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active
26 RUN1	Peripheral control during RUN1 0 Peripheral is frozen with clock gated 1 Peripheral is active
27 RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
28 DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
29 SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
30 TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated 1 Peripheral is active
31 RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

56.3.2.25 Low-Power Peripheral Configuration Registers (ME_LP_PC*n*)

These registers configure eight different types of peripheral behavior during non-run modes.

Offset: 0x0A0 + *n**0x4 (n = 0 to 7) Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STANDBY0	0	0	STOP0	0	HALT0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1170. Low-Power Peripheral Configuration Registers (ME_LP_PC*n*)

Table 1181. ME_LP_PC*n* field descriptions

Field	Description
STANDBY0	Peripheral control during STANDBY0 0 Peripheral is frozen with clock gated 1 Peripheral is active
STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
HALT0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated 1 Peripheral is active

56.3.2.26 Peripheral Control Registers (ME_PCTL*n*)

These registers select the configurations during run and non-run modes for each peripheral. Please see *Table 1170* for information on which ME_PCTL*n* locations are actually occupied. The unoccupied locations contain a read-only byte value of 0x00.

Note: After modifying any of the ME_RUN_PCn, ME_LP_PCn, and ME_PCTLn registers, software must request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behavior between the peripheral clock control process and the clock status reporting in the ME_PS*n registers.*

Offset : 0x0C0 + *n**0x1 (*n* = 0 to 255) Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0(1)	LP_CFG			RUN_CFG		
W								
Reset	0	0	0	0	0	0	0	0

1. This bit is writable but has no impact.

Figure 1171. Peripheral Control Registers (ME_PCTL*n*)

Table 1182. ME_PCTL*n* field descriptions

Field	Description
2:4 LP_CFG	Peripheral configuration select for non-run modes These bits associate a configuration as defined in the ME_LP_PC0–7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5:7 RUN_CFG	Peripheral configuration select for run modes These bits associate a configuration as defined in the ME_RUN_PC0–7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

56.3.2.27 Core Status Register (ME_CS)

This register provides the status of the additional cores.

Offset: 0x1C0 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	S_CORE4	0	0	0	S_CORE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1172. Core Status Register (ME_CS)

Table 1183. ME_CS field descriptions

Field	Description
27 S_CORE4	Core 4 status This bit specifies the current status of the HSM which is controlled by the ME_CCTL4 register. 0 the HSM is disabled 1 the HSM is running
31 S_CORE0	Core 0 status This bit specifies the current status of peripheral core_2 which is controlled by the ME_CCTL0 register. 0 peripheral core_2 is disabled 1 peripheral core_2 is running

56.3.2.28 Core Control Register 0 (ME_CCTL0)

This register controls whether peripheral core_2 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1).

Offset: 0x1C4 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0

Figure 1173. Core Control Register 0 (ME_CCTL0)

Table 1184. ME_CCTL0 field descriptions

Field	Description
2 STANDBY0	Core control during STANDBY0 The peripheral core 2 is always disabled during STANDBY0.
5 STOP0	Core control during STOP0 peripheral core_2 is always disabled during STOP0.
7 HALT0	Core control during HALT0 peripheral core_2 is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running
9 RUN2	Core control during RUN2 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running

Table 1184. ME_CCTL0 field descriptions (continued)

Field	Description
10 RUN1	Core control during RUN1 0 peripheral core_2 is disabled with clock gated 1 peripheral core_2 is running
11 RUN0	Core control during RUN0 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
12 DRUN	Core control during DRUN 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running **Note:** The value of the ME_CCTL*n*[DRUN] bit just before entering STANDBY0 mode is captured by the MC_RGM module and copied for all the run mode bits (TEST, SAFE, DRUN, and RUN*x*) on STANDBY0 exit. For instance: if the ME_CCTL*n*[DRUN] = '1' before entering STANDBY0 mode then the value of all bits ME_CCTL*n*[TEST, SAFE, DRUN and RUN*x*] will be '1' on STANDBY0 exit.
13 SAFE	Core control during SAFE 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
14 TEST	Core control during TEST 0 peripheral core_2 is frozen with clock gated 1 peripheral core_2 is running
15 RESET	Core control during RESET peripheral core_2 is always disabled during RESET.

56.3.2.29 Core Control Register (ME_CCTL4)

This register controls whether the HSM is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1).

Offset: 0x1CC Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1174. Core Control Register (ME_CCTL4)

Table 1185. ME_CCTL4 field descriptions

Field	Description
2 STANDBY0	Core control during STANDBY0 The HSM is always disabled during STANDBY0.
5 STOP0	Core control during STOP0 the HSM is always disabled during STOP0.
7 HALT0	Core control during HALT0 the HSM is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 the HSM is disabled with clock gated 1 the HSM is running
9 RUN2	Core control during RUN2 0 the HSM is disabled with clock gated 1 the HSM is running
10 RUN1	Core control during RUN1 0 the HSM is disabled with clock gated 1 the HSM is running
11 RUN0	Core control during RUN0 0 the HSM is frozen with clock gated 1 the HSM is running
12 DRUN	Core control during DRUN 0 the HSM is frozen with clock gated 1 the HSM is running **Note:** The value of the ME_CCTL*n*[DRUN] bit just before entering STANDBY0 mode is captured by the MC_RGM module and copied for all the run mode bits (TEST, SAFE, DRUN, and RUN*x*) on STANDBY0 exit. For instance: if the ME_CCTL*n*[DRUN] = ‘1’ before entering STANDBY0 mode then the value of all bits ME_CCTL*n*[TEST, SAFE, DRUN and RUN*x*] will be ‘1’ on STANDBY0 exit.
13 SAFE	Core control during SAFE 0 the HSM is frozen with clock gated 1 the HSM is running
14 TEST	Core control during TEST 0 the HSM is frozen with clock gated 1 the HSM is running
15 RESET	Core control during RESET the HSM is always disabled during RESET.

56.3.2.30 Core Address Register 0 (ME_CADDR0)

This register gives the boot address for peripheral core_2 and a bit for controlling whether peripheral core_2 is to be reset on the next mode change that has peripheral core_2 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1). A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted.

Note: The reset value of the ME_CADDR0[ADDR] field is determined by the chip configuration and boot mode.

Offset: 0x1E0 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR														0	RMC
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	0

Figure 1175. Core Address Register 0 (ME_CADDR0)

Table 1186. ME_CADDR0 field descriptions

Field	Description
0:29 ADDR	Core Address This field is used by peripheral core_2 as the boot address (32-bit word aligned) when peripheral core_2 next exits reset.
31 RMC	Reset on Mode Change peripheral core_2 will be reset on the next mode change that has peripheral core_2 configured to be running in the target mode. 0 peripheral core_2 will not be reset on the next mode change 1 peripheral core_2 will be reset on the next mode change that has peripheral core_2 configured to be running in the target mode

56.3.2.31 Core Address Register 4 (ME_CADDR4)

This register gives the boot address for the HSM and a bit for controlling whether the HSM is to be reset on the next mode change that has the HSM configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1). A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted.

Note: The reset value of the ADDR field of the ME_CADDR4 is determined by the chip configuration.

Offset: 0x1F0 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR														0	RMC
W																
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0

Figure 1176. Core Address Register 4 (ME_CADDR4)

Table 1187. ME_CADDR4 field descriptions

Field	Description
0:29 ADDR	Core Address This field is used by the HSM as the boot address (32-bit word aligned) when HSM core next exits reset.
31 RMC	Reset on Mode Change The HSM will be reset on the next mode change that has the HSM configured to be running in the target mode. 0 the HSM will not be reset on the next mode change 1 the HSM will be reset on the next mode change that has the HSM configured to be running in the target mode

56.4 Functional description

56.4.1 Mode transition request

The transition from one mode to another mode is normally handled by software by accessing the mode control register ME_MCTL. But in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

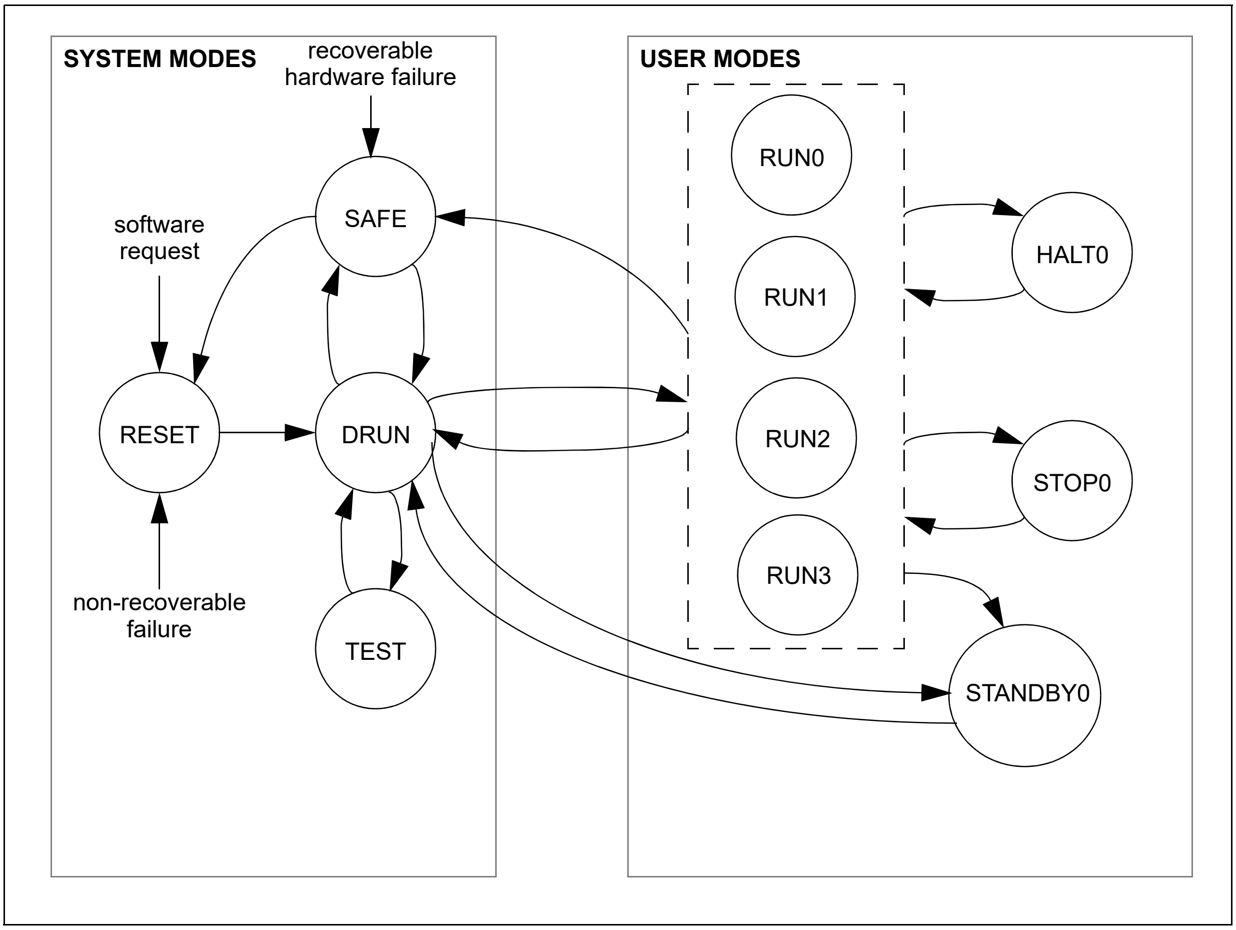
Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<mode>_MC register.The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please see *Section 56.4.5: Mode Transition Interrupts*.

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as: RUN0–3 → RUN0–3, DRUN → DRUN, SAFE → SAFE, and TEST → TEST are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set till the status in the ME_GS register matches the configuration programmed in the respective ME_<mode>_MC register.

Note: It is recommended that software polls the S_MTRANS bit in the ME_GS register after requesting a transition to HALT0, STOP0, or STANDBY0 modes.

If CMU reveals a loss of PLL/XOSC clock before a transition request, such transition is not completed. If failure removal and CMU flags are reset, then transition is executed correctly.

Figure 1177. MC_ME Mode Diagram



56.4.2 Modes details

56.4.2.1 RESET mode

The chip enters this mode on the following events:

- from SAFE, DRUN, RUN0–3, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with either “0000” for a ‘functional’ reset or “1111” for a ‘destructive’ reset.
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details).

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the ME_RESET_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

56.4.2.2 DRUN mode

The chip enters this mode on the following events:

- automatically from RESET mode after completion of the reset sequence
- from RUN0–3, SAFE, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0011”
- from the STANDBY0 mode after an external wakeup event or internal wakeup alarm

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the Flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz int. RC osc. as the system clock. The clock source configuration except system clock source 0 can be chosen by preloaded information in Flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs
- to execute small routines in a ‘ping-pong’ with the STANDBY0 mode

When this mode is entered from STANDBY0 after a wakeup event, the ME_DRUN_MC register content is restored to its pre-STANDBY0 values, and the mode starts in that configuration.

Note: Software must ensure that the code executes from RAM before changing to this mode if the Flash is configured to be in the low-power or power-down state in this mode.

56.4.2.3 SAFE mode

The chip enters this mode on the following events:

- from any mode except RESET when the TARGET_MODE bit field of the ME_MCTL register is written with “0010”
- from any mode except RESET due to a SAFE mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see *Chapter 51: Reset Generation Module (MC_RGM)*, for details)

Note: If a hardware SAFE mode request occurs during RESET, depending on the timing of the SAFE mode request, SAFE mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after DRUN entry. The SAFE mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a SAFE mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

If the SAFE mode is requested by hardware, a SAFE mode interrupt is generated. See *Section 56.4.5.3* for details.

If the SAFE mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the SAFE mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

Note: If software requests to change to the SAFE mode and then requests to change back to the parent mode before the mode transition is completed, the chip's final mode after mode transition will be the SAFE mode, and an invalid mode transition interrupt will be generated.

As long as a SAFE event is active, the system remains in the SAFE mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
  - re-initialize the chip via the DRUN mode, or
  - completely reset the chip via the RESET mode.

Internal weak pull-ups of outputs are enabled upon entering this mode, in case that the PDO bit of the ME_SAFE_MC register is set. The input levels remain unchanged.

56.4.2.4 TEST mode

The chip enters this mode on the following event:

From the DRUN mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0001”

As soon as the above event has occurred, a TEST mode transition request is generated. The mode configuration information for this mode is provided by the ME_TEST_MC register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the SYSCLK bit field to “1111”, and in this case, the only way to exit this mode is via a chip reset.

This mode is intended to be used by software

- to execute software test routines

Note: Software must ensure that the code executes from RAM before changing to this mode if the Flash is configured to be in the low-power or power-down state in this mode.

56.4.2.5 RUN0–3 modes

The chip enters one of these modes on the following events:

- from the DRUN, SAFE, or another RUN0–3 mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0100–0111”
- from the HALT0 mode due to an interrupt event
- from the STOP0 mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a RUN0–3 mode transition request is generated. The mode configuration information for these modes is provided by the ME_RUN0–3_MC registers. In these modes, the Flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

Note: Software must ensure that the code executes from RAM before changing to this mode if the Flash is configured to be in the low-power or power-down state in this mode.

56.4.2.6 HALT0 mode

The chip enters this mode on the following event:

From one of the RUN0–3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1000”.

As soon as the above event has occurred, a HALT0 mode transition request is generated. The mode configuration information for this mode is provided by ME_HALT0_MC register. This mode is quite configurable, and the ME_HALT0_MC register should be programmed according to the system needs. The Flash can be put in low-power or power-down mode as needed. If there is a HALT0 mode request while an interrupt request is active, the transition to HALT0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the core clocks frozen
- the additional core clocks frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (that is within a few system clock cycles of an interrupt event)

The normal exit of the HALT0 mode is from an interrupt event. Wakeup events will not cause HALT0 mode exit.

Note: It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on HALT0 mode exit to ensure that the previous RUN0–3 mode configuration has been fully restored before executing critical code.

56.4.2.7 STOP0 mode

The chip enters this mode on the following event:

From one of the RUN0–3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1010”.

As soon as the above event has occurred, a STOP0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP0_MC register. This mode is fully configurable, and the ME_STOP0_MC register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- the primary PLL
- the secondary PLL

The Flash can be put in power-down mode as needed. If there is a STOP0 mode request while any interrupt or wakeup event is active, the transition to STOP0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- the core clock frozen
- the additional core clocks frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (for example, allow for system clock source to be re-started)

The normal exit from the STOP0 mode is by an interrupt or a wakeup event.

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the STOP0 mode, the 16 MHz internal RC oscillator clock is selected as the system clock until the target clock is available.

Note: It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on STOP0 mode exit to ensure that the previous RUN0–3 mode configuration has been fully restored before executing critical code.

56.4.2.8 STANDBY0 Mode

The chip enters this mode on the following event:

From the DRUN or one of the RUN0–3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1101”.

As soon as the above event occurs, a STANDBY0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STANDBY0_MC register. In this mode, the power supply is turned off for most of the chip. The only parts of the chip that are still powered during this mode are pads mapped on wakeup lines and power domain #0 which contains the MC_RGM, MC_PCU, WKPU, PASS, OSC32k, RTC, LPROSC, SSWU, ADC-STDBY, PMC_DIG. The IRC can be optionally switched off. This is the lowest power consumption mode possible on the chip.

This mode is intended as an extreme low-power mode with

- the cores, the Flashes, and almost all peripherals and memories powered down and to be used by software
- to wait until it is required to do something with no need to react quickly (such as, allow for system power-up and system clock source to be re-started)

The exit sequence of this mode is similar to the reset sequence, and the resets to all but power domain #0 are asserted with timing that is the same as reset PHASE1 through PHASE3. Device will boot from the address specified on ME_CADDR0 register. In case that boot from RAM is requested, ME_CADDR0 will address standby RAM (see *Table 6: System and stand-by RAM memory map*). In this case, it is also possible to keep Flash in power-down by writing ‘01’ to FLAON field in the ME_DRUN_MC register.

If there is a STANDBY0 mode request while any wakeup event is active, the chip mode does not change.

The IO pins in STANDBY mode can be divided in 2 types:

- Not Low Power pin (NLP)
  They are not active in STANDBY and when STANDBY is entered, both input and output buffers inside the pins are disabled. Thus also internal pull-up/down are disabled.

Floating state will not affect the power consumption because of isolation within the pad is automatically applied.

- Low Power pin (LP)
  The LP pins are active during STANDBY. They are configured in CMOS level logic and this configuration cannot be changed. Moreover, when the device enters the STANDBY mode, the pad-keeper feature is activated for LP pins. It means that:
  - if the pad voltage level is above the pad keeper high threshold (PK_h), a weak pull-up resistor is automatically enabled;
  - if the pad voltage level is below the pad keeper low threshold (PK_l), a weak pull-down resistor is automatically enabled.

For the pad-keeper high/low thresholds please refer to the device datasheet. Regarding the pad-keeper weak pull-up/down, these are the same as the ones configurable through SIUL2_MSCR register in running mode, and their values (Rwpu/Rwpd) are showed in the datasheet.

Refer to the SPC584Bx pinout Microsoft Excel file (package sheets) for the list of LP pins.

56.4.3 Mode transition process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

Since the MC_ME ensures that all resources are in the proper state during the mode transition process, the following, for example, do not need to be checked by software.

- When the XOSC is enabled in a mode configuration register, a mode transition to that mode will not complete until the End Of Count timeout occurs for the oscillator. Software does not need to interrogate the End Of Count timeout status bit.
- When a PLL is enabled in a mode configuration register, a mode transition to that mode will not complete until the PLL is locked. Software does not need to interrogate the PLL’s lock status bit.

56.4.3.1 Target mode request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the TARGET_MODE bit field is not updated. An optional interrupt can be generated for invalid mode requests. See *Section 56.4.5*, for details.

In the case of mode transitions occurring because of hardware events such as a reset, a SAFE mode request, or interrupt requests and wakeup events to exit from low-power modes, the TARGET_MODE bit field of the ME_MCTL register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit S_MTRANS of the ME_GS register.

A RESET mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The RESET mode request has the highest priority, and the MC_ME is kept in the RESET mode during the entire reset sequence.

The SAFE mode request has the next highest priority after reset. It can be generated either by software via the ME_MCTL register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

56.4.3.2 Target mode configuration loading

On completion of the *Target mode request* step, the target mode configuration from the ME_<target mode>_MC register is loaded to start the resources (such as voltage sources, clock sources, Flash, pads) control process.

An overview of resource control possibilities for each mode is shown in *Table 1188*. A ‘X’ indicates that a given resource is configurable for a given mode.

Table 1188. MC_ME Resource Control Overview

Resource	Mode							
	RESET	**TEST**	**SAFE**	**DRUN**	**RUN0–3**	**HALT0**	**STOP0**	**STANDBY0**
IRC	on	X on	on	on	on	on	on	X on
XOSC	off	X off	off	X off	X off	X off	X off	off
PLL0	off	X off	off	X off	X off	X off	off	off
PLL1	off	X off	off	X off	X off	X off	off	off
FLASH	normal	X normal	normal	X normal	X normal	X low-power	X power-down	power-down
MVREG	on	on	on	on	on	on	on	off
PDO	off	X off	X on	off	off	off	X off	on

56.4.3.3 Peripheral clocks disable

On completion of the *Target mode request* step and the system clock frequency ramp-down of the *System clock switching* step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0–7 and ME_LP_PC0–7, and the peripheral control registers ME_PCTLn

Note: The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is *good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.*

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a SAFE mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The SAFE mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please see *Section 56.4.6: Peripheral Clock Gating*, for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the SAFE mode.

56.4.3.4 Processor low-power mode entry

If, on completion of the *Peripheral clocks disable* step, the mode transition is to the HALT0 mode, the MC_ME requests the processor to enter its halted state. The processor acknowledges its halt state request after completing all outstanding bus transactions.

If, on completion of the *Peripheral clocks disable* step, the mode transition is to the STOP0 or STANDBY0 mode, the MC_ME requests the processor to enter its stopped state. The processor acknowledges its stop state request after completing all outstanding bus transactions.

56.4.3.5 Processor clocks disable

If, on completion of the *Processor low-power mode entry* step, and all processors which are configured in the ME_CCTLn registers to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processors which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes.

Caution: Clocks to the whole chip including the processor and system memory can be disabled in TEST mode.

56.4.3.6 System clock disable

If, on completion of the *Processor clocks disable* step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

56.4.3.7 Clock sources (Main Voltage Regulator Independent) switch-on

On completion of the *Processor low-power mode entry* step, the MC_ME switches on all clock sources, which do not need the main voltage regulator to be on, based on the

<clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

- the 16 MHz internal RC oscillator: in HALT0 mode; in STOP0 mode; in STANDBY0 mode depending on ME_STANDBY0_MC.IRCON bit setting
- the external crystal oscillator: in HALT0 mode and STOP0 mode, depending on ME_HALT0_MC.XOSCON bit and ME_STOP0_MC.XOSCON bit setting correspondingly
- the primary PLL
- the secondary PLL

Note: Clock sources which need the main voltage regulator to be stable are not controlled by this step.

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the S_<clock source> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order not to disturb the system clock which might require one of these clocks before switching to a different target clock.

56.4.3.8 Main Voltage Regulator Switch-On

On completion of the *Target mode request* step, if the main voltage regulator needs to be switched on from its off state based on the MVRON bit of the ME_<current mode>_MC and ME_<target mode>_MC registers, the MC_ME requests the MC_PCU to power-up the regulator and waits for the output voltage stable status in order to update the S_MVR bit of the ME_GS register.

This step is required only during the exit of the low-power modes HALT0 and STOP0. In this step, the 16 MHz internal RC oscillator is switched on regardless of the target mode configuration, as the main voltage regulator requires the 16 MHz int. RC osc. during power-up in order to generate the voltage status.

During the STANDBY0 exit sequence, the MC_PCU alone manages the power-up of the main voltage regulator, and the MC_ME is kept in RESET or shut off (depending on the power domain #1 status).

56.4.3.9 Flash memory module switch-on

On completion of the *Main Voltage Regulator Switch-On* step, if the Flash memory needs to be switched to normal mode from its low-power or power-down mode based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, the MC_ME requests the Flash memory to exit from its low-power/power-down mode. When the Flash memory is available for access, the S_FLA bit field of the ME_GS register is updated to “11” by hardware.

Caution: It is illegal to switch the FLASH or the DFLASH from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

56.4.3.10 Clock Sources (Main Voltage Regulator Dependent) Switch-On

On completion of the *Clock sources (Main Voltage Regulator Independent) switch-on* and *Main Voltage Regulator Switch-On*, the MC_ME controls all clock sources, which need the main voltage regulator to be on, based on the <clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

- the primary PLL depending on ME_<target mode>_MC.PLL0ON bit
- the secondary PLL depending on ME_<target mode>_MC.PLL1ON bit

56.4.3.11 Pad outputs-on

On completion of the *Main Voltage Regulator Switch-On* step, if the PDO bit of the ME_<target mode>_MC register is cleared, then

- all pad outputs are enabled to return to their previous state
- slew rate control mechanism is switched on

56.4.3.12 Peripheral clocks enable

Based on the current and target chip modes, the peripheral configuration registers ME_RUN_PC0–7, ME_LP_PC0–7, and the peripheral control registers ME_PCTLn, the MC_ME enables the clocks for selected modules as required. If the value of the PWRLVL field of the ME_<target mode>_MC register is less than that of the ME_<current mode>_MC register's, this step is executed only after the *System clock switching* process is completed.

56.4.3.13 System and processor clocks enable

On completion of the *Flash memory module switch-on* step, the MC_ME enables the non-processor system clock and the clocks of all processors which are configured in the ME_CCTLn registers to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of all processors which are configured in the ME_CCTLn registers to be running in the target mode and which have their RMC bits in the ME_CADDRn registers set to '1'. The core_rst_b outputs remain asserted until the end of the mode transition as indicated by the S_MTRANS bit of the MW_GS register at which time they are deasserted.

If the value of the PWRLVL field of the ME_<target mode>_MC register is different from that of the ME_<current mode>_MC register's, the mode change handshake will not be initiated until after the system clock frequency ramp-down step of the *System clock switching* process is completed.

56.4.3.14 Processor low-power mode exit

On completion of the *System and processor clocks enable* step, the MC_ME requests all processors which are configured in the ME_CCTLn registers to be disabled in the current mode and to be running in the target mode by deasserting the corresponding core_stop bits.

56.4.3.15 System clock switching

Based on the SYSCLK bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz int. RC osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (that is the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the external crystal osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (that is the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the primary PLL (PHI) takes effect only after the S_XOSC bit of the ME_GS register is set by hardware (that is the external crystal oscillator has stabilized).
- The target clock configuration for the secondary PLL takes effect only after the S_PLL0 bit of the ME_GS register is set by hardware (that is the primary PLL has stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with “1111”. This is possible only in the TEST mode. In the STANDBY0 mode, the clock configuration is fixed, and the system clock is automatically forced to ‘0’. .

The current system clock configuration can be observed by reading the S_SYSCLK bit field of the ME_GS register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the *Clock sources (Main Voltage Regulator Independent) switch-on* process has completed if the target system clock source is one of the following:
  - the 16 MHz internal RC oscillator
  - the external crystal oscillator
  - the primary PLL
- the *Peripheral clocks disable* process has completed in order not to change the system clock frequency before peripherals close their internal activities
- the *Peripheral clocks enable* process has completed

An overview of system clock source selection possibilities for each mode is shown in *Table 1189*. A ‘X’ indicates that a given clock source is selectable for a given mode.

Table 1189. MC_ME System Clock Selection Overview

System Clock Source	Mode							
	RESET	TEST	SAFE	DRUN	RUN0–3	HALT0	STOP0	STANDBY0
16 MHz int. RC osc.		X (default)		X (default)	X (default)	X (default)		
external crystal osc.		X		X	X	X		
primary PLL (PHI)		X		X	X	X		

Table 1189. MC_ME System Clock Selection Overview (continued)

System Clock Source	Mode							
	RESET	TEST	SAFE	DRUN	RUN0–3	HALT0	STOP0	STANDBY0
secondary PLL		X		X	X	X		
system clock is disabled		X(1)						

1. disabling the system clock during TEST mode will require a reset in order to exit TEST mode

56.4.3.16 Pad switch-off

If the PDO bit of the ME_<target mode>_MC register is ‘1’ then

- Internal weak pull-ups of outputs are enabled if the target mode is SAFE or TEST.

In STANDBY0 mode, the power sequence driver and all pads except the external reset and those mapped on wakeup or SSWU lines are not powered and therefore high impedance. The wakeup and SSWU line configuration remains unchanged.

This step is executed only after the *Peripheral clocks disable* process has completed.

56.4.3.17 Clock sources switch-off

Based on the chip mode and the <clock source>ON bits of the ME_<mode>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<clock source> of the ME_GS register to ‘0’. The following clock sources switched off at this step:

- the 16 MHz internal RC oscillator
- the external crystal oscillator
- the primary PLL
- the secondary PLL

For the clock sources related to the system clock, this step is executed only after the *System clock switching* process has completed.

56.4.3.18 Flash memory switch-off

Based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the Flash is to be put in its low-power or power-down mode, the MC_ME requests the Flash to enter the corresponding power mode and waits for the Flash to acknowledge. The exact power mode status of the Flash is updated in the S_FLA bit field of the ME_GS register. This step is executed only when the *Processor clocks disable* and *System clock disable* processes have completed.

56.4.3.19 Main Voltage Regulator Switch-Off

Based on the MVRON bit of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the main voltage regulator is to be switched off, the MC_ME requests it to power down and clears the availability status bit S_MVR of the ME_GS register.

This step is executed only after completing the following processes:

- *Clock sources switch-off*
- *Flash memory switch-off*

If the target mode is STANDBY0, the main voltage regulator is not switched off by the MC_ME and the STANDBY0 request is asserted after the above processes have completed upon which the MC_PCU takes control of the main regulator. As the MC_PCU needs the 16 MHz int. RC osc., the 16 MHz internal RC oscillator remains active until all the STANDBY0 steps are executed by the MC_PCU after which it may be switched off depending on the IRCON bit of the ME_STANDBY0_MC register.

56.4.3.20 Current mode update

The current mode status bit field S_CURRENT_MODE of the ME_GS register is updated with the target mode bit field TARGET_MODE of the ME_MCTL register when :

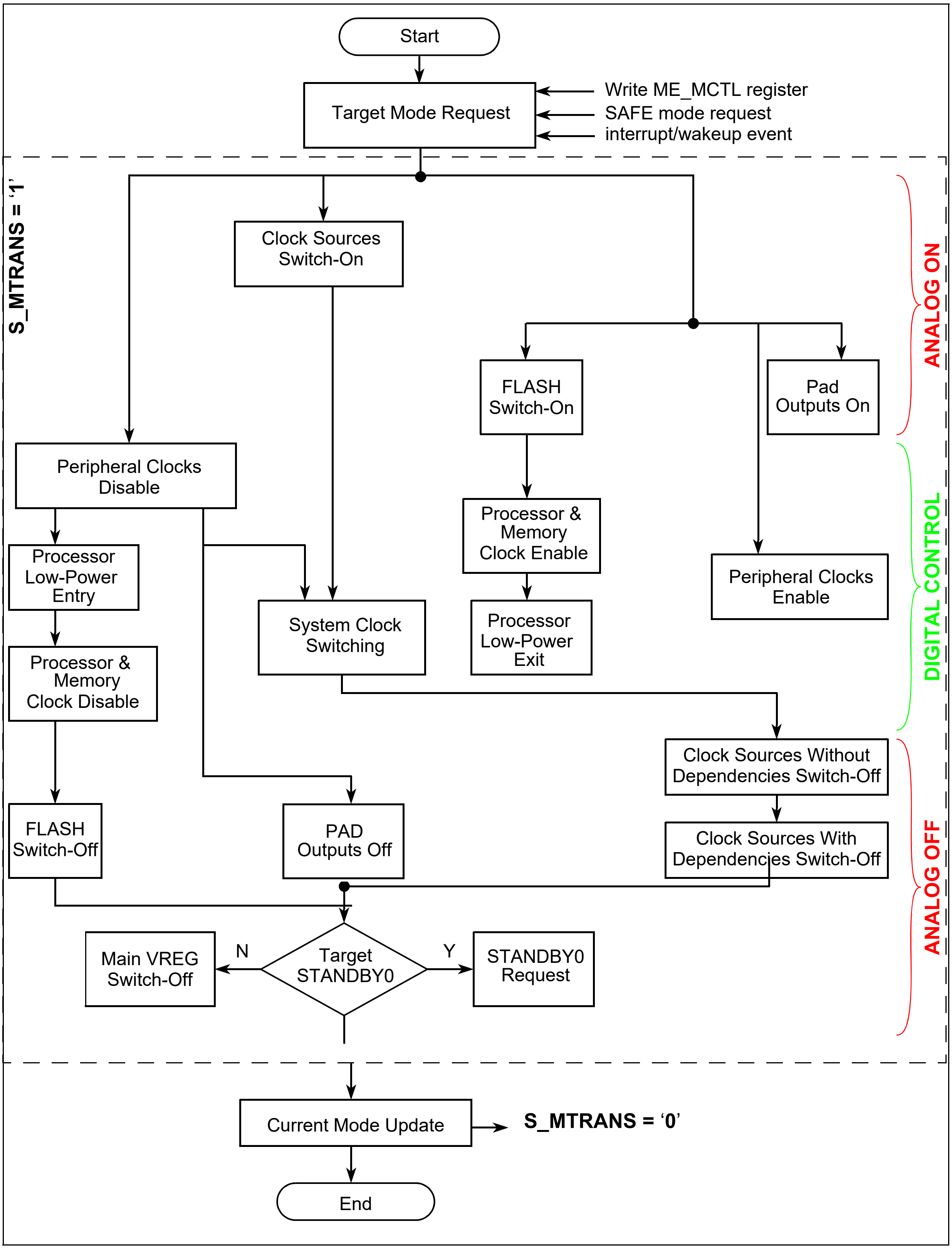
- all the updated status bits in the ME_GS register match the configuration specified in the ME_<target mode>_MC register.
- power sequences are done.
- clock disable/enable process is finished.
- processor low-power mode (stop) entry and exit processes are finished.

Note: SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the ME_GS.S_RC bit to be set. This is to ensure that the SAFE mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the S_MTRANS bit of the ME_GS register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the ME_DMTS register can indicate which process is still in progress.

Figure 1178. MC_ME Transition Diagram



56.4.4 Protection of Mode Configuration Registers

While programming the mode configuration registers ME_<mode>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the 16 MHz int. RC osc. is selected as the system clock, IRC must be on.
- If the external crystal osc. clock is selected as the system clock, XOSC must be on.
- If the primary PLL (PHI) clock is selected as the system clock, PLL0 must be on.
- If the secondary PLL clock is selected as the system clock, PLL1 must be on.

Note: Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration “00” for the FLAON bit field is reserved.
- System clock configurations marked as ‘reserved’ may not be selected.
- Configuration “1111” for the SYSCLK bit field is allowed for the TEST mode and STANDBY mode, and only in this case may all system clock sources be turned off.

Caution: If the system clock is stopped during TEST mode, the chip can exit only via a system reset.

56.4.5 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a SAFE mode transition not due to a software request, and indicating when a mode transition has completed.

56.4.5.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the **ME_<mode>_MC** registers violating the protection rules mentioned in *Section 56.4.4*, the interrupt pending bit I_ICONF of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF of the ME_IM register is ‘1’.

In addition, during a mode transition, if a clock source has been configured in the ME_<target mode>_MC register to be off and a peripheral requiring this clock source to be on has been enabled via the ME_RUN_PC0–7/ME_LP_PC0–7 and ME_PCTLn registers, the interrupt pending bit I_ICONF_CU of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CU of the ME_IM register is ‘1’.

If an attempt to write to one of the ME_CADDRn registers is made after a mode change request has been made via the second write to the ME_CTL register and before the completion of that mode transition as indicated by the S_MTRANS bit in the ME_GS register deasserting, the interrupt pending bit I_ICONF_CC of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CC of the ME_IM register is ‘1’.

56.4.5.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the SAFE mode and the SAFE mode request from MC_RGM is active, and if the target mode requested is other than RESET or SAFE, then this new mode request is considered to be invalid, and the S_SEA bit of the ME_IMTS register is set.

- If the TARGET_MODE bit field of the ME_MCTL register is written with a value different from the specified mode values (that is a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the S_NMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If some of the chip modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the S_DMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit S_MRI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition to a mode other than SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is ‘1’), the mode transition illegal status bit S_MTI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a new mode request occurs while a mode transition to the SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is ‘1’), the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a HALT0 mode request occurs while an interrupt request is active, the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a STOP0 mode request occurs while a wakeup or interrupt request is active, the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.

Note: As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA, S_MRI, and S_MTI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the RESET or SAFE mode irrespective of the mode transition status.
- As the exit of HALT0 and STOP0 modes depends on the interrupts of the system which can occur at any instant, these requests to return to RUN0–3 modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined ‘reasonable’ amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.

Warning: Depending on the cause of the mode transition not completing, aborting via a request back to the parent mode may not be able to recover. In this case, software should request a change to either SAFE or RESET mode, since the cause of the hang-up is most likely due to a hardware fault.

- Self-transition requests (for example, RUN0 → RUN0) are not considered as invalid even when the mode transition process is active (that is S_MTRANS is ‘1’). During the low-power mode exit process, if the system is not able to enter the respective RUN0–3 mode properly (that is all status bits of the ME_GS register match with configuration bits in the ME_<mode>_MC register), then software can only request the SAFE or RESET mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit I_IMODE of the ME_IS register is set, and an interrupt request is generated if the mask bit M_IMODE of the ME_IM register is ‘1’.

56.4.5.3 SAFE Mode Transition Interrupt

Whenever the system enters the SAFE mode as a result of a SAFE mode request from the MC_RGM due to a hardware failure, the interrupt pending bit I_SAFE of the ME_IS register is set, and an interrupt is generated if the mask bit M_SAFE of ME_IM register is ‘1’.

The SAFE mode interrupt pending bit can be cleared only when the SAFE mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a SAFE mode request). If the system is already in SAFE mode, any new SAFE mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the SAFE mode interrupt pending bit is not set when the SAFE mode is entered by a software request (by programming the ME_MCTL register).

56.4.5.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (that is the S_MTRANS bit of ME_GS register transits from ‘1’ to ‘0’), the interrupt pending bit I_MTC of the ME_IS register is set, and an interrupt request is generated if the mask bit M_MTC of the ME_IM register is ‘1’. The interrupt bit I_MTC is not set when entering low-power modes HALT0 and STOP0 in order to avoid the same event requesting the immediate exit of these low-power modes.

56.4.6 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0–7 are chosen only during the software running modes DRUN, TEST, SAFE, and RUN0–3. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTLn registers.

The low-power peripheral configuration registers ME_LP_PC0–7 are chosen only during the low-power modes HALT0, STOP0, and STANDBY0. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTLn registers.

Any modifications to the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTLn registers do not affect the clock gating behavior until a new mode transition request is generated.

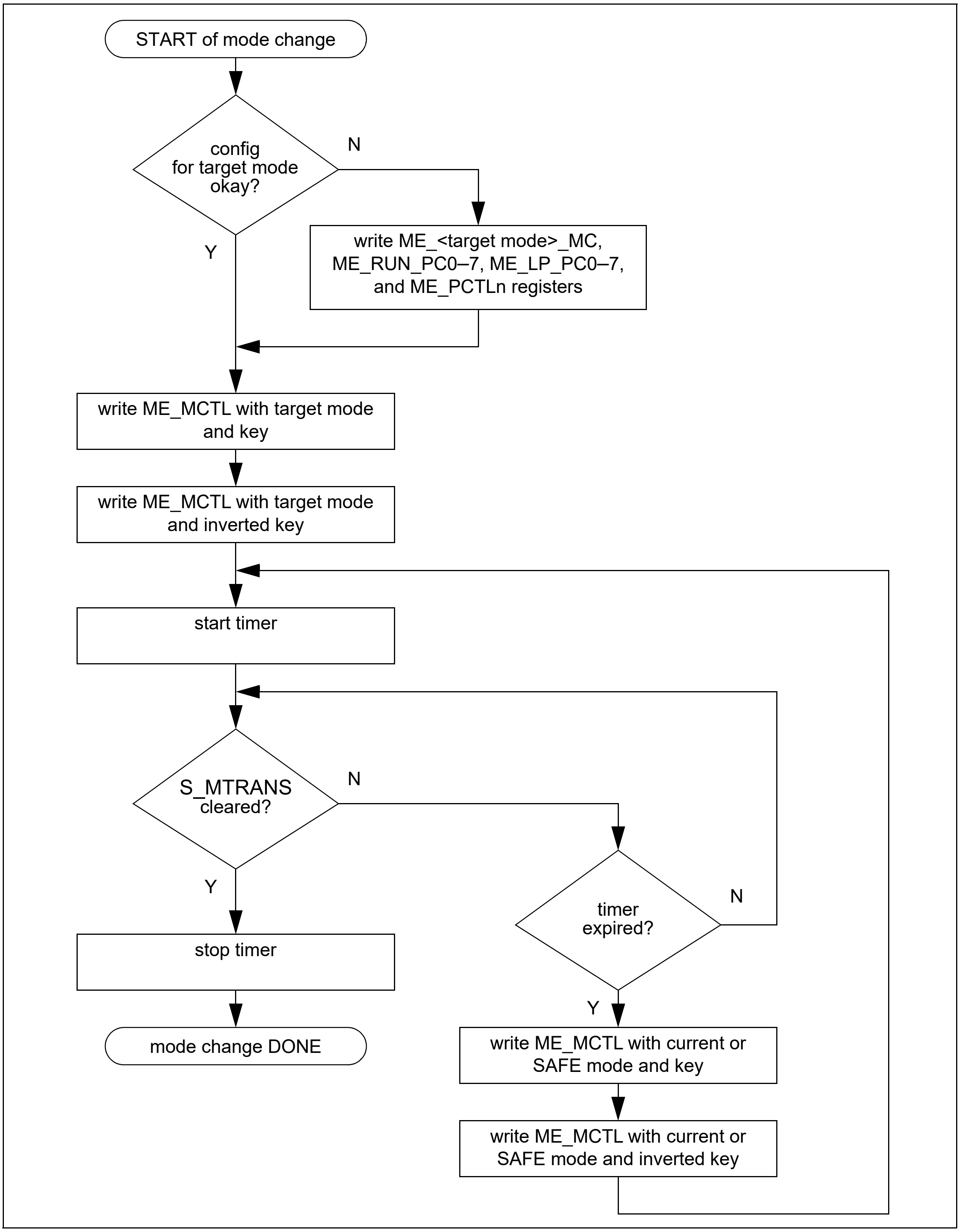
Whenever the processor enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTLn register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits. Any further modifications of the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTLn registers during a debug session will take effect immediately without requiring any new mode request.

56.4.7 Application Example

Figure 1179 shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

Figure 1179. MC_ME Application Example Flow Diagram



57 e200z420n3 Core Debug Support

This chapter describes the debug features of the e200z420n3 core.

57.1 Overview

Internal debug support in the e200z420n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware-based debugger, which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *PowerISA 2.06*. e200z420n3 supports a subset of these defined facilities. In addition to the facilities defined in *PowerISA 2.06*, e200z420n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

57.1.1 Software debug facilities

e200z420n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors that are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is “owned” by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

57.1.1.1 *PowerISA 2.06* compatibility

The e200z420n3 core implements a subset of the *PowerISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

57.1.2 Additional debug facilities

In addition to the debug functionality defined in *PowerISA 2.06*, e200z420n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger (Refer to *Section 57.1.4, Sharing debug resources by software/hardware*). A data trace port is also provided.

57.1.2.1 Data trace port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z420n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (that is the normal setting). Refer to the “Data Trace Control Register (DTC)” section in the Core (e200z420n3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

57.1.3 Hardware debug facilities

The e200z420n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z420n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the “hardware-owned” debug facilities. In addition, since resources are “owned” by the hardware debugger, inconsistent values may be present if software attempts to read “hardware-owned” debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in *Section 57.3, Debug registers* are reserved for use by the external debugger. The same events described in *Section 57.2, Software debug events and exceptions* are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via

EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

57.1.4 Sharing debug resources by software/hardware

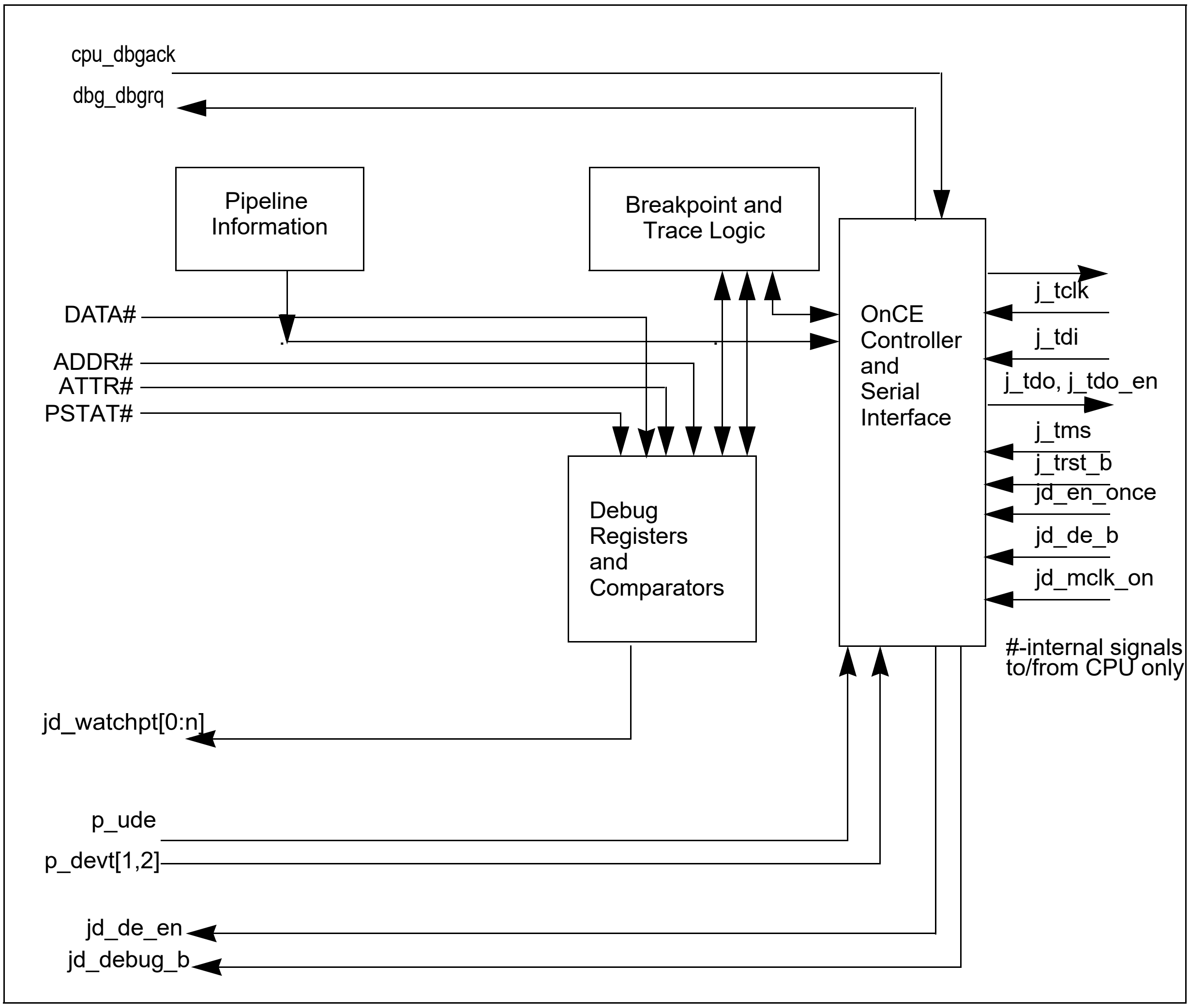
Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources that set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources that set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

57.1.4.1 Simultaneous hardware and software debug event handing

Since it is possible that a “hardware-owned” resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented that guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

Figure 1180 shows the e200z420n3 debug resources.

Figure 1180. e200z420n3 debug resources



57.2 Software debug events and exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource that is “owned” by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to ‘1’.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction that enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** that causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *PowerISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z420n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2, which are described in *Section 57.2.11, External debug event*.
- The Performance Monitor Interrupt event PMI, which is described in *Section 57.2.13, Performance Monitor Interrupt debug event*.

The e200z420n3 debug configuration supports most of these event types. Unsupported *PowerISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported.

A brief description of each of the event types follows.

57.2.1 Instruction Address Compare event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

57.2.2 Data Address Compare event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note: In contrast to the PowerISA 2.06 definition, Data Address Compare events on e200z420n3 do not prevent the load or store class instruction from completing. If a load or store class instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC\_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, that is no data comparisons are performed since these instructions do not have associated data values.

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

*DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.*

*DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.*

*Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.*

57.2.2.1 Data Address Compare event status updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC\_OFST}$ field when the DVC DAC status is recorded.

Table 1190 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in *Table 1190*, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 1190. DAC events and resultant updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.

Table 1190. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.
DVC DACx	DVC DACy, Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.
DVC DACx	DVC DACy, Normal Ldst	Ldst instruction, no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.
DVC DACx	DVC DACy, Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy, Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.
DVC DACx	DVC DACy, Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.

Table 1190. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	Any instruction (no exception)	DSI, with or without DAC, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3’b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction (no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.

Table 1191 through *Table 1194* show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 1191. DAC events and resultant updates, dual-issue case 1

Instruction sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3’b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)

Table 1191. DAC events and resultant updates, dual-issue case 1 (continued)

Instruction sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	**Event(s)**	**Result**
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b100. No ESR update. DSRR0 points to instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.

Table 1192. DAC events and resultant updates, dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3’b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3’b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b101. DSRR0 points to instruction (7). No ESR update. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the “no DACy” case.

Table 1192. DAC events and resultant updates, dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	**Event(s)**	**Result**
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1193. DAC events and resultant updates, dual-issue case 3

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.

Table 1193. DAC events and resultant updates, dual-issue case 3 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	**Event(s)**	**Result**
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1194. DAC events and resultant updates, dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.

Table 1194. DAC events and resultant updates, dual-issue case 4 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

57.2.3 Linked Instruction Address Compare and Data Address Compare events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note: Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

57.2.4 Trap debug event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

57.2.5 Branch Taken debug event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ cannot be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

57.2.6 Instruction Complete debug event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to ‘1’ while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to ‘1’ while MSR_{DE}=1.

57.2.7 Interrupt Taken debug event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

57.2.8 Critical Interrupt Taken debug event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

57.2.9 Return debug event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (that is before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

57.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (that is before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

57.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to ‘1’ to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

57.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to ‘1’ to record the debug exception. This debug event is an asynchronous event.

57.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to ‘1’ to record the debug exception. This debug event is an asynchronous event.

57.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not “owned” by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not “owned” by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not “owned” by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers are performed.

57.3.1 Debug Address and Value registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in *Figure 1181*) corresponds to the byte data value transferred to/from memory byte offset such as 0, 8, and the least significant byte of the register (labeled B7 in *Figure 1181*) corresponds to byte offset such as 7, F. When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison; they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z420n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

B0	B1	B2	B3
0–7	8–15	16–23	24–31
B4	B5	B6	B7
32–39	40–47	48–55	56–63

SPR 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset: Unaffected

Figure 1181. DVC1, DVC2 registers

57.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z420n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

57.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z420n3 adds some implementation specific bits to this register, as seen in *Figure 1182*.

0	1	2–3	4	5	6	7	8	9	10	11	12–13	14–15	16	17	18	19	20	21	22	23–24	25	26	27–31
EDM	IDM	RST	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC2	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	0	CIRPT	CRET	0

SPR - 308; Read/Write; Reset(1) - 0x0

Figure 1182. Debug Control Register 0 (DBCR0) register

1. DBCR0$_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if DBCR0$_{EDM}$=0, as well as unconditionally by **m_por**. If DBCR0$_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b,** and only software-owned resources indicated by EDBRAC0 and the DBCR0$_{RST}$ field will be reset by **p_reset_b**. The DBCR0$_{RST}$ field will always be reset by **p_reset_b** regardless of the value of DBCR0$_{EDM}$.

Table 1195 provides bit definitions for Debug Control Register 0.

Table 1195. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0.
		Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to ‘1’ before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to ‘1’, all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR$_{DE}$=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR$_{DE}$ was cleared, will cause a Debug interrupt.

Table 1195. DBCR0 field descriptions (continued)

Bit	Name	Description
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled

Table 1195. DBCR0 field descriptions (continued)

Bit	Name	Description
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1 IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

57.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in *Figure 1183*.

Bits	0–1	2–3	4–5	6–7	8–9	10–15	16–17	18–19	20–21	22–23	24–25	26–31
Field	IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	0	IAC3US	IAC3ER	IAC4US	IAC4ER	IAC34M	0

SPR - 309; Read/Write; Reset(1) - 0x0

Figure 1183. DBCR1 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.**

Table 1196 provides bit definitions for Debug Control Register 1.

Table 1196. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1

Table 1196. DBCR1 field descriptions (continued)

Bit	Name	Description
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

57.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in *Figure 1184*.

DAC1US		DAC1ER		DAC2US		DAC2ER		DAC12M		DAC1LNK	DAC2LNK	DVC1M		DVC2M		DVC1BE								DVC2BE							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 310; Read/Write; Reset(1) - 0x0

Figure 1184. DBCR2 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1197 provides bit definitions for Debug Control Register 2.

Table 1197. DBCR2 field descriptions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.

Table 1197. DBCR2 field descriptions (continued)

Bit	Name	Description
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies *Exact Address Compare* since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR.
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **Note:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **Note:** Inactive byte lanes of the memory access are automatically masked.

Table 1197. DBCR2 field descriptions (continued)

Bit	Name	Description
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **Note:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **Note:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1*xxxxxxx* Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. *x*1*xxxxxx* Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. *xx*1*xxxxx* Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. *xxx*1*xxxx* Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. *xxxx*1*xxx* Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. *xxxxx*1*xx* Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. *xxxxxx*1*x* Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. *xxxxxxx*1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1*xxxxxxx* Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. *x*1*xxxxxx* Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. *xx*1*xxxxx* Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. *xxx*1*xxxx* Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. *xxxx*1*xxx* Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. *xxxxx*1*xx* Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. *xxxxxx*1*x* Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. *xxxxxxx*1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

57.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in *Figure 1185*.

0	1	2	3	4–11	12	13	14	15	16–19	20–23	24–27	28–31
0	DVC1C	0	DVC2C	0	0	DAC1XMH	0	DAC2XMH	DAC1XM	DAC2XM	DAC1CFG	DAC2CFG

SPR - 563; Read/Write; Reset(1) - 0x0

Figure 1185. DBCR4 register

1. DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1198 provides bit definitions for Debug Control Register 4.

Table 1198. DBCR4 field descriptions

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. Refer to details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. Refer to details in the DBCR2 register definition 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits

Table 1198. DBCR4 field descriptions (continued)

Bit	Name	Description
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.

Table 1198. DBCR4 field descriptions (continued)

Bit	Name	Description
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01*xx* Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01*xx* Reserved 1*xxx* Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

57.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in *Figure 1186*

0–1	2–3	4–5	6–7	8–9	10–15	16–17	18–19	20–21	22–23	24–25	26–31
IAC5US	IAC5ER	IAC6US	IAC6ER	IAC56M	0	IAC7US	IAC7ER	IAC8US	IAC8ER	IAC78M	0

SPR - 564; Read/Write; Reset(1) - 0x0

Figure 1186. DBCR5 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1199 provides bit definitions for Debug Control Register 5.

Table 1199. DBCR5 field descriptions

Bit	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved

Table 1199. DBCR5 field descriptions (continued)

Bit	Name	Description
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

57.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in *Figure 1187*.

0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
IAC1XM	IAC2XM	IAC3XM	IAC4XM	IAC5XM	IAC6XM	IAC7XM	IAC8XM

SPR - 603; Read/Write; Reset(1) - 0x0

Figure 1187. DBCR6 register

1. DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1200 provides bit definitions for Debug Control Register 6.

Table 1200. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 = Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Control. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

Table 1200. DBCR6 field descriptions (continued)

Bit	Name	Description
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101 – 1111 = Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

57.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in *Figure 1188*.

Bits	0–11	12	13	14	15	16–19	20–23	24–27	28–31
Field	0	0	DAC3XMH	0	DAC4XMH	DAC3XM	DAC4XM	DAC3CFG	DAC4CFG

SPR - 596; Read/Write; Reset(1) - 0x0

Figure 1188. DBCR7 register

1. DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1201 provides bit definitions for Debug Control Register 7.

Table 1201. DBCR7 field descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High. DAC3XMH extends the range of the DAC3XM field. 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits

Table 1201. DBCR7 field descriptions (continued)

Bit	Name	Description
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001–1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

57.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in *Figure 1189*.

0–1	2–3	4–5	6–7	8–9	10	11	12–31
DAC3US	DAC3ER	DAC4US	DAC4ER	DAC34M	DAC3LNK	DAC4LNK	0

SPR - 597; Read/Write; Reset[(1)] - 0x0

Figure 1189. DBCR8 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1202 provides bit definitions for Debug Control Register 8.

Table 1202. DBCR8 field descriptions

Bit	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 DAC3 debug events not affected by MSR_{PR} 01 Reserved 10 DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC3 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 DAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 DAC4 debug events not affected by MSR_{PR} 01 Reserved 10 DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC4 debug events can only occur if MSR_{PR}=1 (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 DAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1

Table 1202. DBCR8 field descriptions (continued)

Bit	Name	Description
8:9	DAC34M	Data Address Compare 3/4 Mode 00 Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 No effect 1 DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 No effect 1 DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

57.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr** *DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A ‘1’ causes the bit to be cleared, and a ‘0’ has no effect. Debug Status bits are set by Debug events only while Internal Debug mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in *Figure 1190*.

IDE	UDE	MRR		ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST			0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 1190. Debug Status Register (DBSR)

Table 1203 provides bit definitions for the Debug Status Register.

Table 1203. DBSR field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to ‘1’ if an imprecise debug event occurs due to a DAC event on a load or store that is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 No reset occurred since these bits were last cleared by software 01 A hard reset occurred since these bits were last cleared by software 10 Reserved 11 Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred

Table 1203. DBSR field descriptions (continued)

Bit	Name	Description
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

57.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is updated by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in *Figure 1191*.

Data Effective Address
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 600; Read/Write; Reset - unaffected

Figure 1191. Debug Data Effective Address Register (DDEAR)

57.3.3 External Debug Resource Allocation Control (EDBRAC0) register

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources that generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources that generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event that is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event that is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

The EDBRAC0 register is shown in *Figure 1192*.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28–31
0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	DNI	DQM	0

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b**, reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 1192. External Debug Resource Allocation Control (EDBRAC0) register

Table 1204 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 1204. EDBRAC0 field descriptions

Bit	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored. 0 Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable.
2	RST	Reset Field Control 0 $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 Event owned by hardware debug. 1 Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 Event owned by software debug. $DBCR0_{TRAP}$ is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 Event owned by software debug. IAC1 control fields are software readable/writable.

Table 1204. EDBRAC0 field descriptions (continued)

Bit	Name	Description
9	IAC2	Instruction Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields. 1 Event owned by software debug. DAC1 control fields are software readable/writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 Event owned by software debug. DAC2 control fields are software readable/writable.
15	—	Reserved
16	RET	Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 Event owned by software debug. IAC8 control are software readable/writable.

Table 1204. EDBRAC0 field descriptions (continued)

Bit	Name	Description
21	DEVT1	External Debug Event Input 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event 0 Event owned by hardware debug. No **mtpmr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 Event owned by software debug. PMRs are software readable/writable. **Note:** This bit is reset to ‘1’.
24	MPU	Memory Protection Unit Debug Event 0 Event owned by hardware debug. No **mpuwe** access by software to region descriptors that have the DEBUG control bit set to ‘1’ or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** This bit is reset to ‘1’.
25	CIRPT	Critical Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENT_{DQTAG}$ field or DDAM register. Attempted access by software is ignored. 1 $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 1205 shows which resources are controlled by EDBRAC0 settings.

Table 1205. EDBRAC0 resource control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DGM}$	Software Accessible via mtspr, affected by p_reset_b
0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	All debug registers
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{IDM}$
1	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{RST}$
1	1	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{UDE}$
1	1	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{ICMP}$
1	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{BRT}$
1	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{IRPT}$
1	1	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{TRAP}$
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR1_{IAC12M}$

Table 1205. EDBRAC0 resource control (continued)

EDBCR0$_{EDM}$	EDBRAC0$_{IDM}$	EDBRAC0$_{RST}$	EDBRAC0$_{UDE}$	EDBRAC0$_{ICMP}$	EDBRAC0$_{BRT}$	EDBRAC0$_{IRPT}$	EDBRAC0$_{TRAP}$	EDBRAC0$_{IAC1}$	EDBRAC0$_{IAC2}$	EDBRAC0$_{IAC3}$	EDBRAC0$_{IAC4}$	EDBRAC0$_{IAC5}$	EDBRAC0$_{IAC6}$	EDBRAC0$_{IAC7}$	EDBRAC0$_{IAC8}$	EDBRAC0$_{DAC1}$	EDBRAC0$_{DAC34}$	EDBRAC0$_{DAC2}$	EDBRAC0$_{RET}$	EDBRAC0$_{DEVT1}$	EDBRAC0$_{DEVT2}$	EDBRAC0$_{PMI}$	EDBRAC0$_{MPU}$	EDBRAC0$_{CIRT}$	EDBRAC0$_{CRET}$	EDBRAC0$_{DNI}$	EDBRAC0$_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC3, DBCR0$_{IAC3}$, DBCR1$_{IAC3US, IAC3ER}$, DBCR6$_{IAC3XM}$
1	1	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC4, DBCR0$_{IAC4}$, DBCR1$_{IAC4US, IAC4ER}$, DBCR6$_{IAC4XM}$
1	1	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR1$_{IAC34M}$
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC5, DBCR0$_{IAC5}$, DBCR5$_{IAC5US, IAC5ER}$, DBCR6$_{IAC5XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC6, DBCR0$_{IAC6}$, DBCR5$_{IAC6US, IAC6ER}$, DBCR6$_{IAC6XM}$
1	1	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DBCR5$_{IAC56M}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC7, DBCR0$_{IAC7}$, DBCR5$_{IAC7US, IAC7ER}$, DBCR6$_{IAC7XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	IAC8, DBCR0$_{IAC8}$, DBCR5$_{IAC8US, IAC8ER}$, DBCR6$_{IAC8XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	DBCR5$_{IAC78M}$

Table 1205. EDBRAC0 resource control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{GRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DAC3, DAC4 $DBCR7_{DAC\{3,4\}, DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US, DAC\{3,4\}ER, DAC\{3,4\}M}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	1	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC12M}$
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC1LNK}$
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC2LNK}$
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	$DBCR8_{DAC3LNK}$
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	$DBCR8_{DAC4LNK}$

Table 1205. EDBRAC0 resource control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRPT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	$DBCR0_{RET}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	$DBCR0_{DEVT1}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	$DBCR0_{DEVT2}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	All PMRs
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	MPU entries with DEBUG bit set, $MPU0CSR0_{DRDEN, DWDEN,IDEN}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	$DBCR0_{CIRPT}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	$DBCR0_{CRET}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—		—	—	—	—	—	—	1	—	
1	— (1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	$DEVENT_{DQTAG}$, DDAM

1. IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in *Table 1205* are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

57.3.4 Debug Event Select register (DEVENT) register

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a ‘1’ to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. Refer to the “Data Acquisition ID Tag Field” section in the Core (e200z420n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in *Figure 1193*.

DQTAG								0																DEVNT							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR 975; Reset(1): 0x0

Figure 1193. Debug Event Select register (DEVENT) register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.** Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

Table 1206 provides bit definitions for the Debug Event Register.

Table 1206. DEVENT field descriptions

Bit	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 No signal is asserted *xxxxxxx*1 **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock *xxxxxx*1*x* **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock *xxxxx*1*xx* **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock *xxxx*1*xxx* **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock *xxx*1*xxxx* **p_devnt_out[4]** is asserted for one clock *xx*1*xxxxx* **p_devnt_out[5]** is asserted for one clock *x*1*xxxxxx* **p_devnt_out[6]** is asserted for one clock 1*xxxxxxx* **p_devnt_out[7]** is asserted for one clock

57.3.5 Debug Data Acquisition Message (DDAM) register

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. Refer to the “Data Acquisition Messaging” section in the Core (e200z420n3) Nexus 3 Module chapter for details.

The DDAM register is shown in *Figure 1194*.

DDAM
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR 576; Reset[1]: 0x0

Figure 1194. Debug Data Acquisition Message (DDAM) register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1207 provides bit definitions for the Debug Data Acquisition Message Register.

Table 1207. DDAM field descriptions

Bit	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

57.4 Using debug resources for stack limit checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access that generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in *Table 1198*. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to ‘1000’ or the $DBCR8_{DAC3CFG}$ field to ‘1000’ (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to ‘1000’ with $DBCR2_{DAC1US}$ set to ‘00’ or ‘10’, or the $DBCR7_{DAC3CFG}$ field is set to ‘1000’ with $DBCR8_{DAC3US}$ set to ‘00’ or ‘10’, or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to ‘1000’ with $DBCR2_{DAC1US}$ set to ‘00’ or ‘01’, or the $DBCR7_{DAC3CFG}$ field is set to ‘1000’ with $DBCR8_{DAC3US}$ set to ‘00’ or ‘01’, or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in *Table 1198* and *Table 1201* in *Section 57.3.2, Debug Control and Status registers*. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

57.5 External debug support

External debug support is supplied through the OnCE controller serial interface, which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to ‘1’ through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of *Section 57.2, Software debug events and exceptions* refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note: On the initial setting of $EDBCR0_{EDM}$ to ‘1’, other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to ‘1’. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information that could cause unintended operation.

*It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.*

*$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all hardware debug events are blocked. Watchpoints are not blocked.*

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

57.5.1 External debug registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug mode.

57.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in *Figure 1195*.

0	1	2–6	7	8–31
EDM	DNH_EN	0	DNI_CTL	0

Reset[1]: 0x0

Figure 1195. External Debug Control Register 0 (EDBCR0)

1. EDBCR0 is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**.

Table 1208 provides bit definitions for External Debug Control Register 0.

Table 1208. EDBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0.
1	DNH_EN	**dnh** Instruction Enable 0 Execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 Execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	—	Reserved
7	DNI_CTL	**dni** Instruction Control 0 When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSR_{DE}=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. **Note:** This control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1.
8:31	—	Reserved

57.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in *Figure 1196*.

IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST			0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Read/Write; Reset(1) - 0x0000_0000

Figure 1196. External Debug Status Register 0 (EDBSR0)

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$=0.

Table 1209 provides bit definitions for External Debug Status Register 0.

Table 1209. EDBSR0 field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store that is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events that are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred

Table 1209. EDBSR0 field descriptions (continued)

Bit	Name	Description
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH\_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

57.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A ‘1’ stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events that are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in *Figure 1197*.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	0				

Read/Write; Reset(1) - 0x0000_0000

Figure 1197. External Debug Status Register Mask 0 (EDBSRMSK0)

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$=0.

Table 1210 provides bit definitions for External Debug Status Register Mask 0.

Table 1210. EDBSRMSK0 field descriptions

Bit	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved

Table 1210. EDBSRMSK0 field descriptions (continued)

Bit	Name	Description
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

57.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits that are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in *Figure 1198*.

Data Effective Address
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Read/Write; Reset: unaffected

Figure 1198. External Debug Data Effective Address Register (EDDEAR)

57.5.2 OnCE introduction

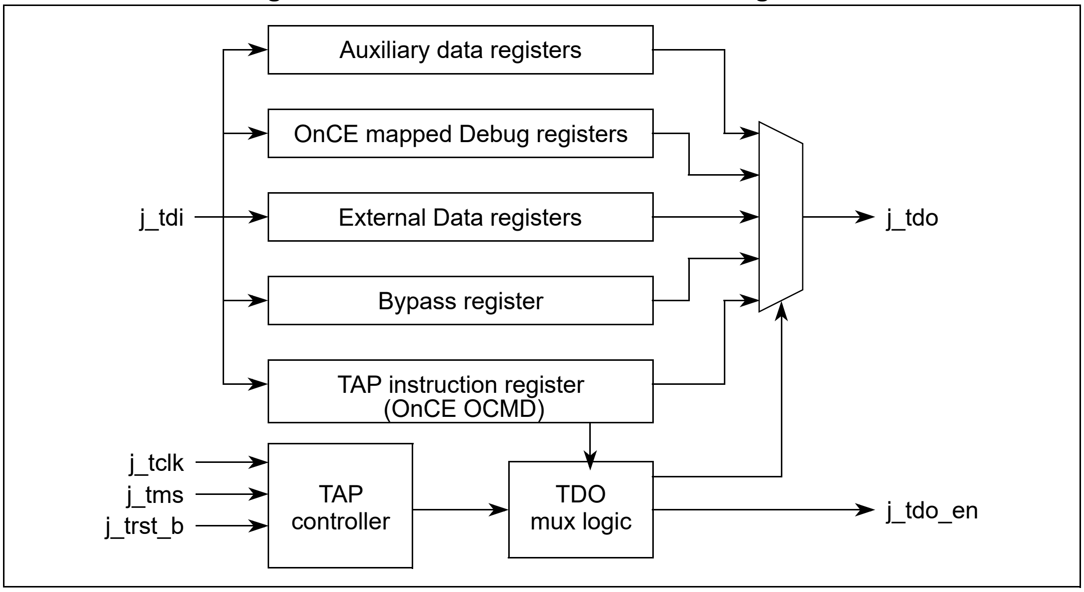
The e200z420n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z420n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

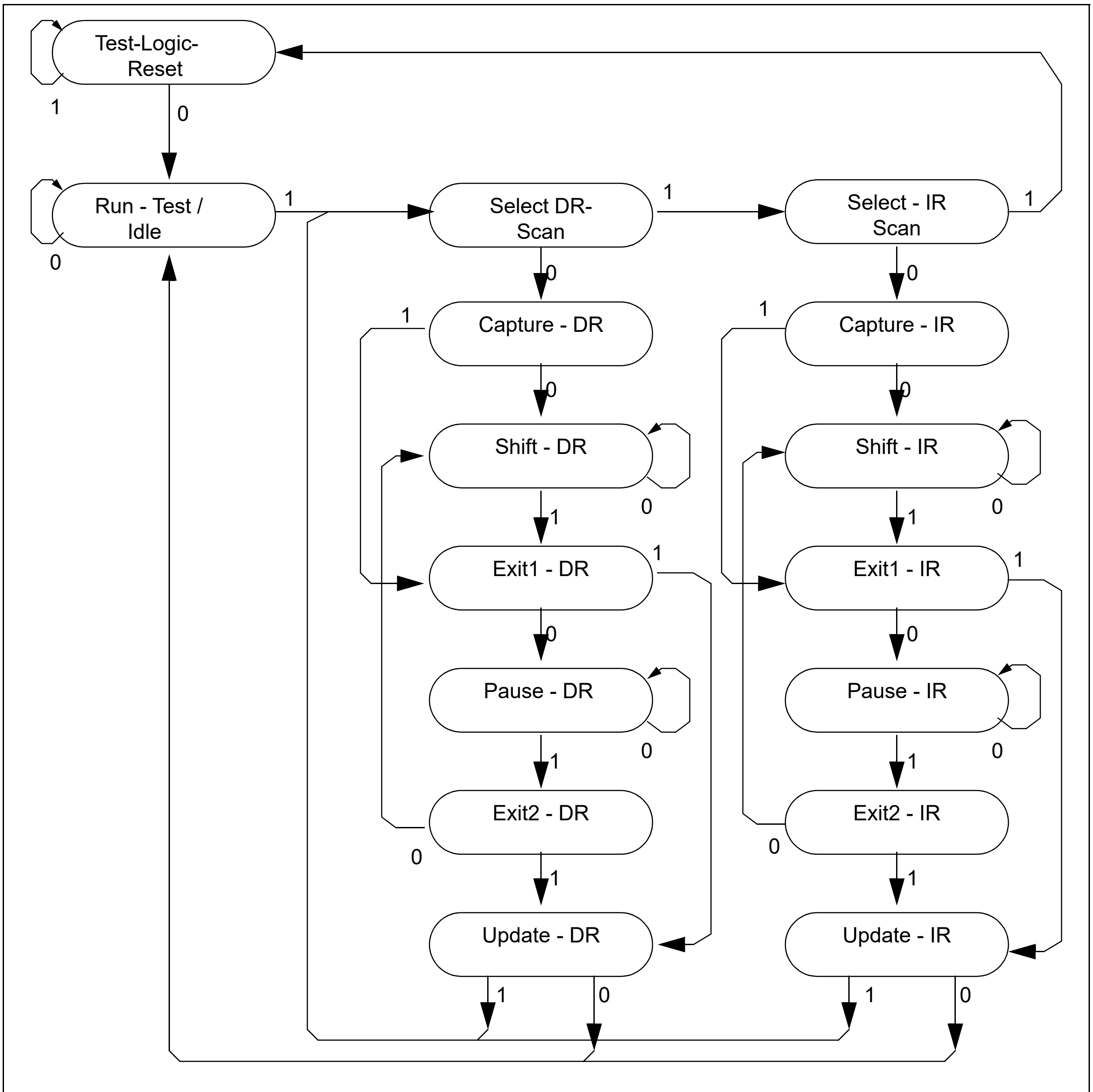
The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

Figure 1199. OnCE TAP controller and registers



The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.

Figure 1200. OnCE controller implementation



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core, which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), *Section 57.5.6.1, OnCE Status register*) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (refer to *Section 57.5.6.2, OnCE Command (OCMD) register*). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

57.5.3 JTAG/OnCE pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

Table 1211 details the primary JTAG/OnCE interface signals.

Table 1211. JTAG/OnCE primary interface signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z420n3) Core Complex Overview.

57.5.4 OnCE internal interface signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

57.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

57.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

57.5.4.3 CPU address and attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

57.5.4.4 CPU data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

57.5.5 OnCE interface signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks that may need information pertaining to debug operation.

57.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a “Bypass” command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/Idle, or Update_DR TAP states. A new value will take effect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

57.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z420n3 interface) provides a fast means of entering the Debug mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug mode. Once Debug mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems that do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

57.5.5.3 OnCE Debug output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control that are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either “No Register Selected” or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. Refer to *Section 57.5.6.2, OnCE Command (OCMD) register* for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

57.5.5.4 CPU Clock On input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU’s **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z420n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

57.5.5.5 Watchpoint events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to *Table 1218* for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

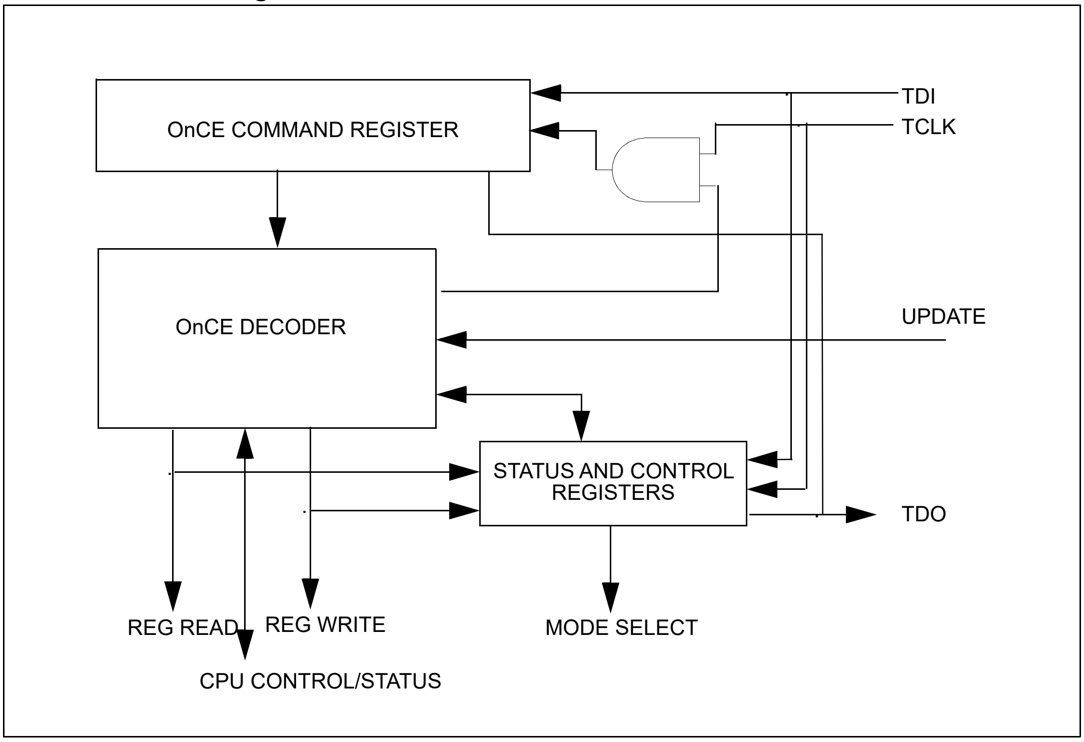
57.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates “no register selected”. This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with “no register selected”, and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs that are not in debug mode will ignore the command.

57.5.6 OnCE controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. *Figure 1201* is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR, and Update-DR states.

Figure 1201. OnCE Controller and Serial Interface



57.5.6.1 OnCE Status register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 1202. OnCE Status register

Table 1212 provides bit definitions for the OnCE Status register.

Table 1212. OnCE Status register field descriptions

Bit	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 Inactive state 1 Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including such as External, Critical, machine check, Storage, Alignment, Program) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
2	—	Reserved, set to zero
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

57.5.6.2 OnCE Command (OCMD) register

The OnCE Command (OCMD) register is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in *Figure 1203*. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step or exit functionality, or both, to be performed, even though the command appears to have no data resource requirement associated with it.

R/W	GO	EX	RS[0:6]						
0	1	2	3	4	5	6	7	8	9

Reset - 10’b1000000010 on assertion of **j_trst_b** or **m_por**, or while in the Test_Logic_Reset state

Figure 1203. OnCE Command (OCMD) register

Table 1213 provides bit definitions for the OCMD.

Table 1213. OCMD field descriptions

Bit	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 Write the data associated with the command into the register specified by RS[0:6] 1 Read the data contained in the register specified by RS[0:6]
1	GO	Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction that resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to “No Register Selected”. Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. **Note:** Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. Refer to *Section 57.5.9.6, Exiting Debug mode and interrupt blocking*. 0 Inactive (no action taken) 1 Execute instruction in IR

Table 1213. OCMD field descriptions (continued)

Bit	Name	Description
2	EX	Exit Command Bit 0 Remain in debug mode 1 Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to “No Register Selected”. Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. **Note:** If the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. **Note:** Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. Refer to *Section 57.5.9.6, Exiting Debug mode and interrupt blocking*.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. *Table 1214* indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 1214 indicates the OnCE register addresses.

Table 1214. OnCE register addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)

Table 1214. OnCE register addressing (continued)

RS[0:6]	Register Selected
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) — *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) — *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)

Table 1214. OnCE register addressing (continued)

RS[0:6]	Register Selected
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3 Access
111 1101	LSRL Select (refer to Test Specification)
111 1110	Enable_OnCE[(1)]
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The OnCE Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

Single-stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (such as interrupts, machine checks, bus error conditions) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of $EDBCR0_{EDM}$. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

57.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z420n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (refer to *Section 57.7, MPU operation during debug*). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in *Figure 1204*.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0								I_DMDIS	0		I_DVLE	I_DI	0			D_DMDIS	0			D_DI	0	D_DG	0						WKUP	FDB	DR

Reset - 0xo000_0000 on **m_por**, **j_trst_b**, or entering Test_logic_Reset state

Figure 1204. OnCE Control Register (OCR)

Table 1215 provides bit definitions for the OnCE Control Register.

Table 1215. OCR field descriptions

Bit	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to‘1’ to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug ‘VLE’ Attribute Bit This bit is used to provide the ‘VLE’ attribute bit to be used during a debug session. **Note:** This bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug ‘I’ Attribute Bit This bit is used to provide the ‘I’ attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to‘1’ to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
17:19	—	Reserved
20	D_DI	Data Side Debug ‘I’ Attribute Bit This bit is used to provide the ‘I’ attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug ‘G’ Attribute Bit This bit is used to provide the ‘G’ attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.

Table 1215. OCR field descriptions (continued)

Bit	Name	Description
30	FDB	Force Breakpoint Debug Mode Bit This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo-instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. **Note:** This bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter Debug mode. The CPU will indicate that Debug mode has been entered via the data scanned out in the shift-IR state. 0 No Debug mode request 1 Unconditional Debug mode request When the DR bit is set, the processor will enter Debug mode at the next instruction boundary.

57.5.7 Access to debug resources

Resources contained in the OnCE Module that do not require the processor core to be halted for access may be accessed while the e200z420n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z420n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z420n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (such as DBSR) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

Table 1216 provides a list of access requirements for OnCE registers.

Table 1216. OnCE register access requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for write access	Requires CPU to be halted for read access	Requires CPU to be halted for write access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*(1)	
DBCR0	Y	Y	Y	N	*(1)	*EDBCR0 $_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*(1)	
DEVENT	Y	Y	Y	N	*(1)	
EDBRAC0 (DBERC0)	Y	N	Y	N	*(1)	
DBSR	Y	Y	Y	N(2)	*(1)	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*(1)	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	

Table 1216. OnCE register access requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for write access	Requires CPU to be halted for read access	Requires CPU to be halted for write access	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	—	—	—	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

57.5.8 Methods of entering Debug mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input. Refer to *Section 57.5.5.1, OnCE Enable (jd_en_once)* for details.

57.5.8.1 External debug request during RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b**, and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z420n3 core to enter Debug mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z420n3 core does not execute an instruction before entering Debug mode, although the first instruction to be executed will be fetched prior to entering Debug mode in order to properly initialize the CPUSCR.

57.5.8.2 Debug request during RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

57.5.8.3 Debug request during normal activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

57.5.8.4 Debug request during Waiting, Halted, or Stopped state

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** output during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

57.5.8.5 Software request during normal activity

Upon executing a '**bkpt'** pseudo-instruction (for e200z420n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt'** pseudo-instruction has entered the instruction register.

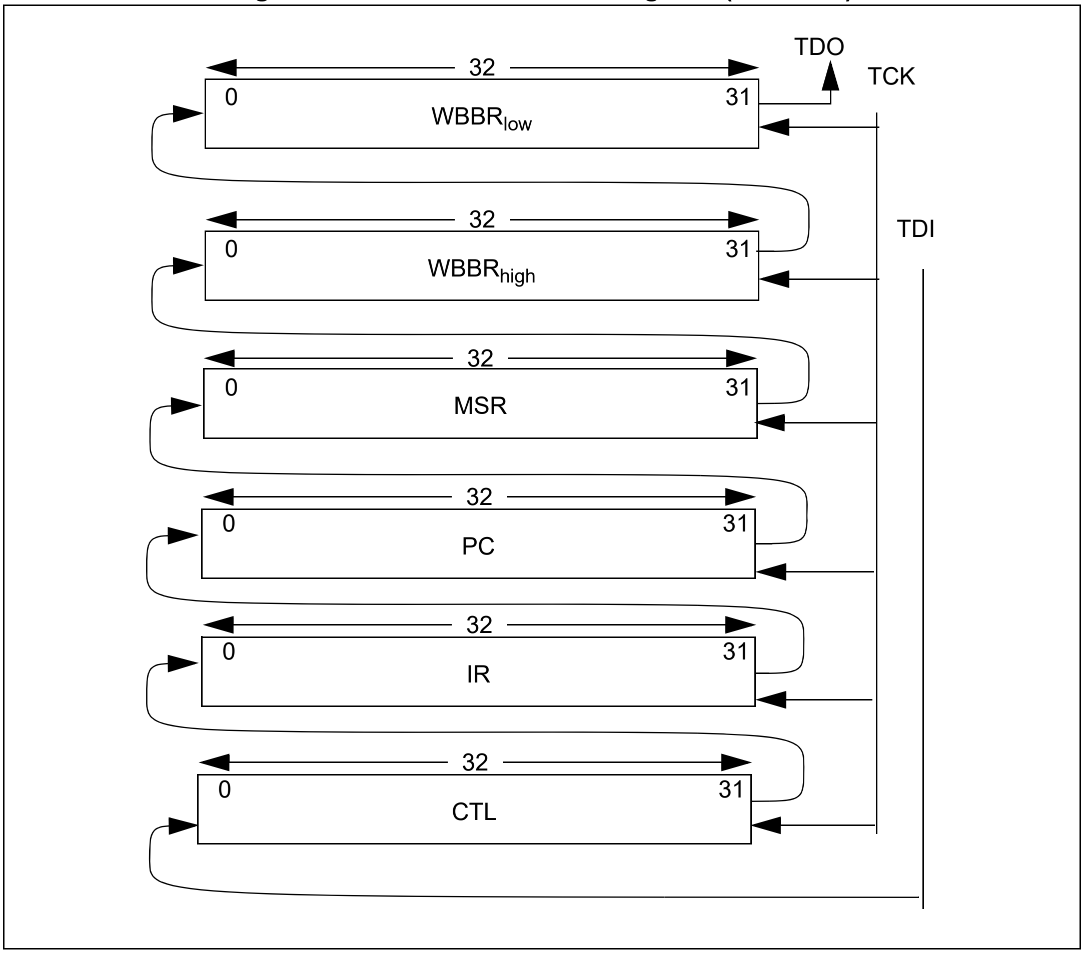
57.5.8.6 Debug notify halt instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH\_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z420n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

57.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. *Figure 1205* shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

Figure 1205. CPU Scan Chain Register (CPUSCR)



57.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

57.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

Bits	0–9	10	11	12	13	14	15	16–19	20	21	22	23	24	25	26	27	28	29	30	31
Field	*	IRSTAT14	IRSTAT13	IRSTAT12	IRSTAT11	IRSTAT10	WAITING	PCOFST	PCINV	FFRA	IRSTAT0	IRSTAT1	IRSTAT2	IRSTAT3	IRSTAT4	IRSTAT5	IRSTAT6	IRSTAT7	IRSTAT8	IRSTAT9

Figure 1206. Control State Register (CTL)

Table 1217. CTL field descriptions

Bit	Name	Description
0:9	*	Internal State Bits These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. Refer to *Section 57.5.5.3, OnCE Debug output (jd_debug_b)*, which is not part of the normal program execution flow, these bits should be set to a 0.
10	IRSTAT14	IR Status Bit 14 This control bit indicates an MPU Instruction Address Compare event status for the IR. 0 No MPU Instruction Address Compare event occurred on the fetch of this instruction. 1 An MPU Instruction Address Compare event occurred on the fetch of this instruction.
11	IRSTAT13	IR Status Bit 13 This control bit indicates an Instruction Address Compare 8 event status for the IR. 0 No Instruction Address Compare 8 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 8 event occurred on the fetch of this instruction.
12	IRSTAT12	IR Status Bit 12 This control bit indicates an Instruction Address Compare 7 event status for the IR. 0 No Instruction Address Compare 7 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 7 event occurred on the fetch of this instruction.
13	IRSTAT11	IR Status Bit 11 This control bit indicates an Instruction Address Compare 6 event status for the IR. 0 No Instruction Address Compare 6 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 6 event occurred on the fetch of this instruction.
14	IRSTAT10	IR Status Bit 10 This control bit indicates an Instruction Address Compare 5 event status for the IR. 0 No Instruction Address Compare 5 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 5 event occurred on the fetch of this instruction.
15	WAITING	WAITING State Status This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit. 0 CPU was not in the waiting state when debug mode was entered 1 CPU was in the waiting state when debug mode was entered

Table 1217. CTL field descriptions (continued)

Bit	Name	Description
16:19	PCOFST	PC Offset Field This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (Refer to PCINV, which overrides this field). 0000 No correction required. 0001 Subtract 0x04 from PC. 0010 Subtract 0x08 from PC. 0011 Subtract 0x0C from PC. 0100 Subtract 0x10 from PC. 0101 Subtract 0x14 from PC. All other encodings are reserved
20	PCINV	PC and IR Invalid Status Bit This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered. 0 No error condition exists. 1 Error condition exists. PC and IR are corrupted.
21	FFRA	Feed Forward RA Operand Bit This control bit causes the content of the WBBRlo to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.For most LSP instructions using rA||rB as a 64-bit source operand, WBBRhi, lo is used to supply the 64-bit source value. This allows the debug firmware to update processor registers — initialize the WBBRlo with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register. Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent. 0 No action. 1 Content of $WBBR_{lo}$ used as operand value.
22	IRSTAT0	IR Status Bit 0 This control bit indicates a TEA status for the IR. 0 No TEA occurred on the fetch of this instruction. 1 TEA occurred on the fetch of this instruction.
23	IRSTAT1	IR Status Bit 1 This control bit is reserved.

Table 1217. CTL field descriptions (continued)

Bit	Name	Description
24	IRSTAT2	IR Status Bit 2 This control bit indicates an Instruction Address Compare 1 event status for the IR. 0 No Instruction Address Compare event 1 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 1 occurred on the fetch of this instruction.
25	IRSTAT3	IR Status Bit 3 This control bit indicates an Instruction Address Compare 2 event status for the IR. 0 No Instruction Address Compare 2 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 2 event occurred on the fetch of this instruction.
26	IRSTAT4	IR Status Bit 4 This control bit indicates an Instruction Address Compare 3 event status for the IR. 0 No Instruction Address Compare event 3 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 3 occurred on the fetch of this instruction.
27	IRSTAT5	IR Status Bit 5 This control bit indicates an Instruction Address Compare 4 event status for the IR. 0 No Instruction Address Compare 4 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 4 event occurred on the fetch of this instruction.
28	IRSTAT6	IR Status Bit 6 This control bit indicates a Parity Error status for the IR. 0 No Parity Error occurred on the fetch of this instruction. 1 Parity Error occurred on the fetch of this instruction from the I-Cache or the IMEM.
29	IRSTAT7	IR Status Bit 7 This control bit indicates a Precise External Termination Error status for the IR. 0 No Precise External Termination Error occurred on the fetch of this instruction. 1 A Precise External Termination Error occurred on the fetch of this instruction.
30	IRSTAT8	IR Status Bit 8 This control bit indicates the PowerISA VLE status for the IR. 0 IR contains a BookE instruction. (unused encoding) 1 IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit. **Note:** This bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).
31	IRSTAT9	IR Status Bit 9 This control bit is reserved.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bits set to ‘0’, and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit that was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC + 4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events that may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to ‘0’s if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to ‘0’ to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be reentered prior to any further instruction execution.

57.5.9.3 Program Counter (PC) register

The PC is a 32-bit register that stores the value of the program counter that was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

57.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed, which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to *Section 57.5.9.2, Control State Register (CTL)* for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions that do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

57.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

57.5.9.6 Exiting Debug mode and interrupt blocking

When exiting debug mode with a Go+Exit, “asynchronous” interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will reenter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

57.5.10 Reserved registers (Reserved)

The reserved registers are used to control various test control logic. These registers are not intended for customer use. To preclude device or system damage, or both, these registers should not be accessed.

57.6 Watchpoint support

e200z420n3 supports the generation and signalling of watchpoints when operating in internal debug mode ($DBCR0_{IDM}$=1) or in external debug mode ($EDBCR0_{EDM}$=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $EDBCR0_{EDM}$ or $DBCR0_{IDM}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31])** for various development control purposes (refer to the “Watchpoint Trace Messaging” section in the Core (e200z420n3) Nexus 3 Module chapter).

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

Table 1218 shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 1218. Watchpoint output signal assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]

Table 1218. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever **p_pm_event** transitions from a ‘0’ to a ‘1’ while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on $PMLCa0_{PMP}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on $PMLCa1_{PMP}$
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on $PMLCa2_{PMP}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on $PMLCa3_{PMP}$
jd_watchpt[27]	MPU	Memory Protection Unit watchpoint Asserted whenever a the MPU generates a debug event based on region descriptor matches with DEBUG control enabled.

Table 1218. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

57.7 MPU operation during debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR *Section 57.5.6.3, OnCE Control Register (OCR)* for more detail.

Normal operation of the MPU may be modified during a ‘debug session’ via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (*Section 57.5.6.3, OnCE Control Register (OCR)* for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

57.8 Cache array access during debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

57.9 Basic steps for enabling, using, and exiting External Debug mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug mode and initializing debug registers:

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to ‘1’. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set.

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 57.5.9.2, Control State Register (CTL)*), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with “no-register selected”, Go cleared, and determines that the PCU has reentered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 57.5.9.2, Control State Register (CTL)*), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should rewrite the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 57.5.9.2, Control State Register (CTL)*), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note: These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

58 Debug and Calibration Interface (DCI)

58.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

To provide high-speed calibration capability, the DCI also provides a mechanism to use 5-pin LFAST debug tool which shares its pins with the standard JTAG interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features for the Production device (PD) only.

58.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- Port sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

58.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Production device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using LFAST or software debug mode.
  - When using LFAST with a connected calibration/debug tool, JTAG packets are received from the tool via LFAST and output JTAG packets are transmitted back to the tool. The LFAST’s 5-pin interface consists of an LVDS pair for transmit signals, an LVDS pair of receive signals and a reference clock output signal.
  - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode. The LFAST-based debug interface shares its five pins with the standard JTAG pins.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

As the LFAST-based debug interface shares its five pins with JTAG pins, the DCI provides a safe switching logic to switch to LFAST-based debug mode.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as SPU. The DCI also provides the Event Out (EVTO) signal to the debug tool in case of the occurrence of any internal debug event.

Figure 1207 shows the DCI block diagram.

Figure 1207. DCI block diagram

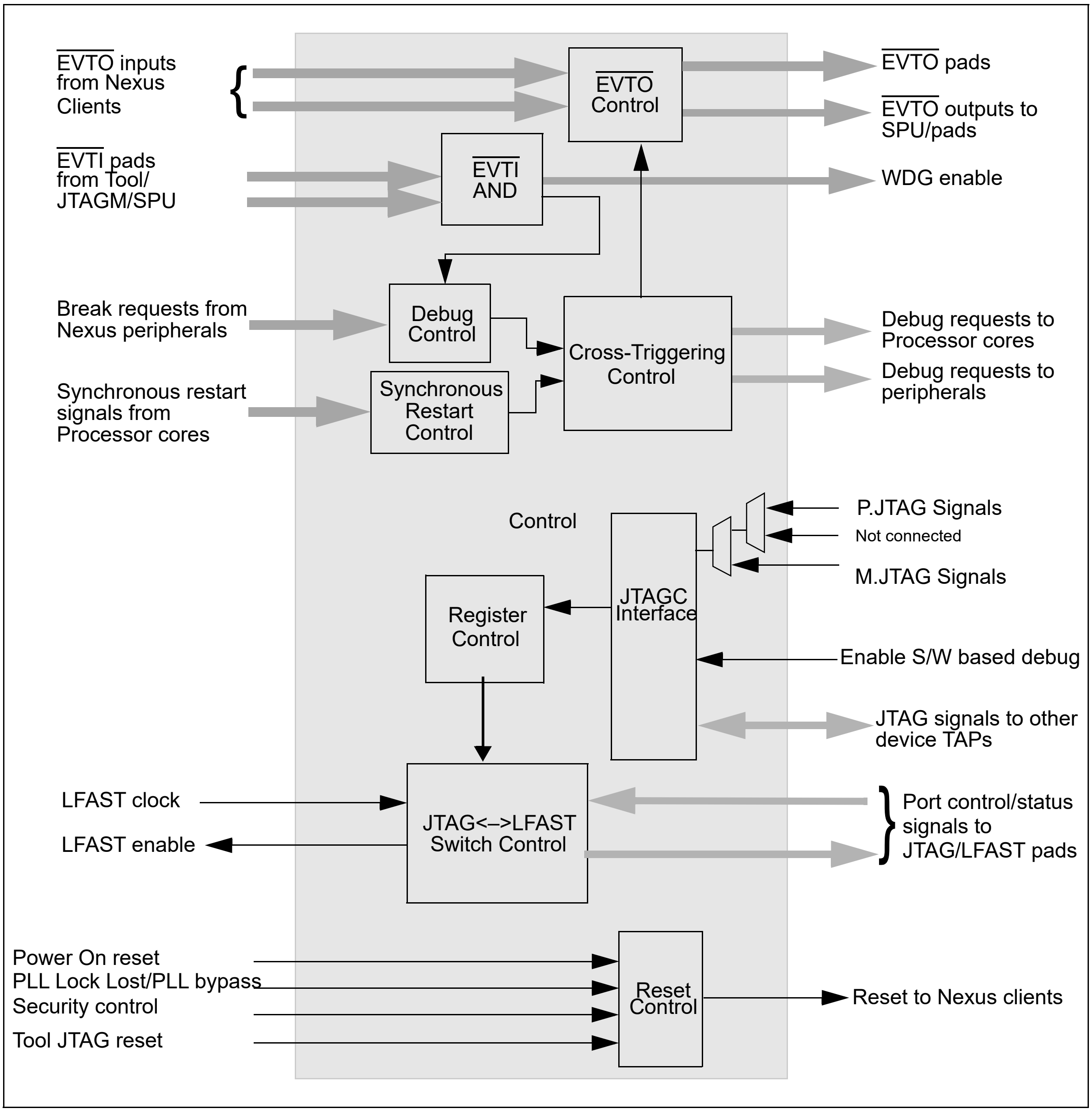
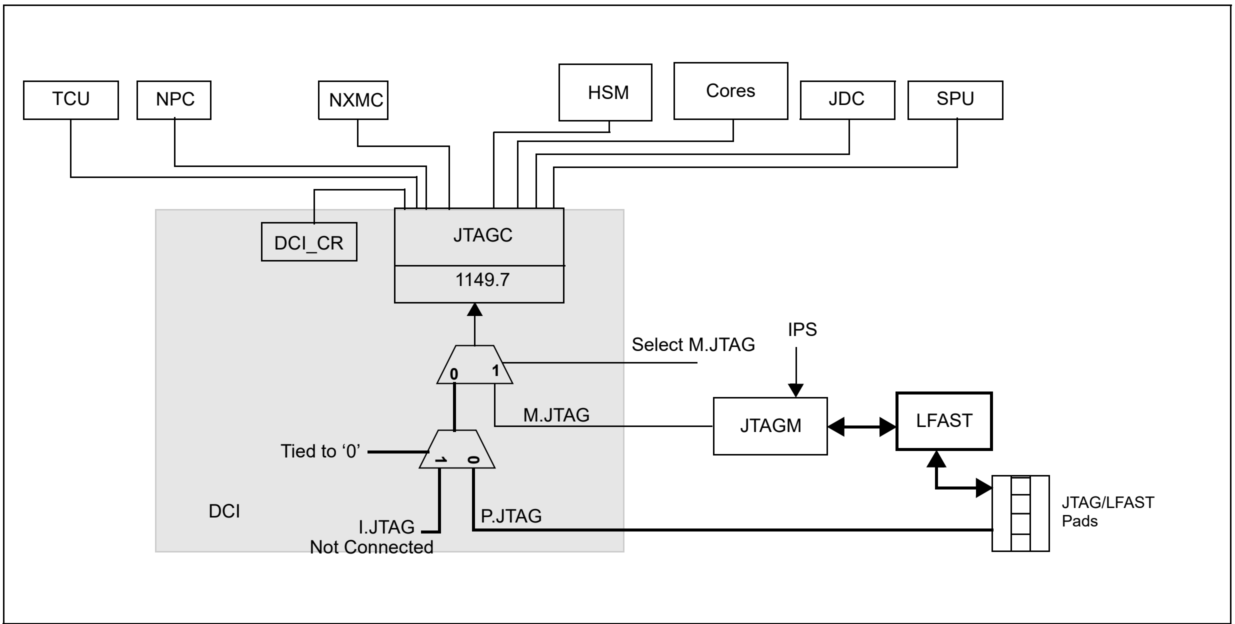


Figure 1208 shows the DCI configuration for the standard production package.

Figure 1208. DCI in a standard configuration



58.1.3 Operating modes

Selection between various JTAG sources is done using a JTAG register bit, DCI_CR[EN_LFAST], and a JTAGM register bit, JTAGM_MCR[DTM]. The JTAG bit, DCI_CR[EN_LFAST], can only be programmed using JTAG instructions and is used to select standard JTAG or LFAST debug tool. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when either DCI_CR[EN_LFAST] or JTAGM_MCR[DTM] is set.

Table 1219 shows a summary of the configuration options.

Table 1219. DCI JTAG source selection

DCI_CR[EN_LFAST]	JTAGM_MCR[DTM]	DCI Source	JCOMP
0	0	P.JTAG (Standard JTAG)	Asserted
1	x	M.JTAG (LFAST)	Asserted
x	1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated.

The M.JTAG provides an alternate interface to the debug logic.

When enabled, it allows high speed debug through the LFAST and JTAGM modules. The high speed link uses LFAST commands that are translated to JTAG commands in the

JTAGM module. The selection between LFAST based debug or software based debug is managed in the JTAGM. However, it is important to note that the dynamic switching between tool-based debug and software debug is not allowed.Software based debug is done only in those cases when an external tool is not present.

An LFAST based tool starts in JTAG mode and then switches to LFAST mode. This switching is managed through a control set of operations which is described in *Section 58.1.3.1: Port sharing between JTAG and LFAST modes on page 1951*. During the switch from JTAG to LFAST and back, the DCI uses a latched value of JCOMP to prevent all debug configuration reset.

The LFAST uses LVDS ports for LFAST communication and the JTAG ports uses CMOS level GPIOs. These signals are double-bonded to share the same pins at the tool interface. *Table 1220* shows the port mapping.

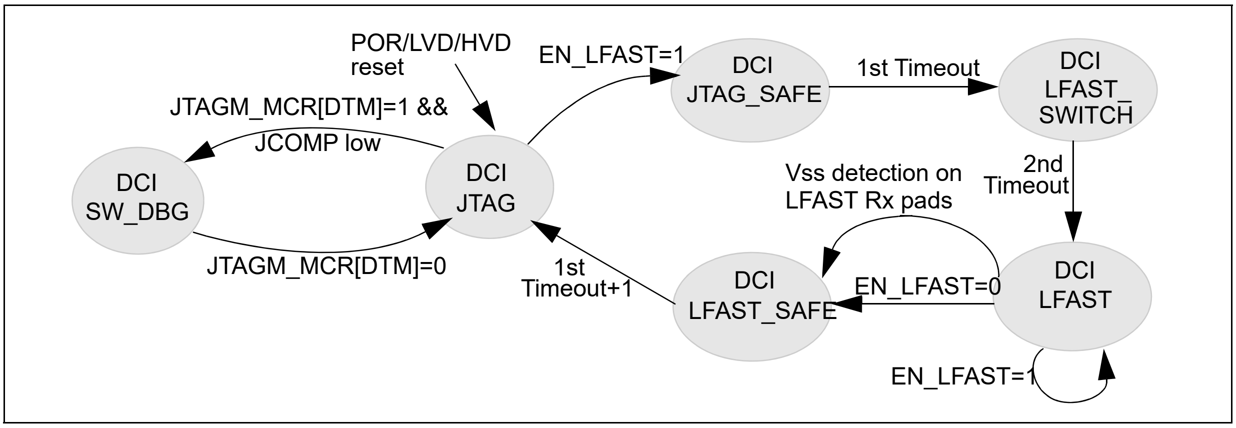
When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

Table 1220. DCI LFAST/JTAG port sharing

JTAG pad	Type in JTAG mode	LFAST pad	Type in LFAST mode
TDI	Input	LVDS TxN	Output
TMS	Input	LVDS TxP	Output
TDO	Output	LVDS RxP	Input
JCOMP	Input	LVDS RxN	Input
TCK	Input	DRCLK	Output

Figure 1209 shows the DCI mode switching operation.

Figure 1209. DCI operating modes state diagram



Note: JTAGM_MCR[jtagm_JCOMP] should not be asserted in LFAST mode to avoid any undesired behavior.

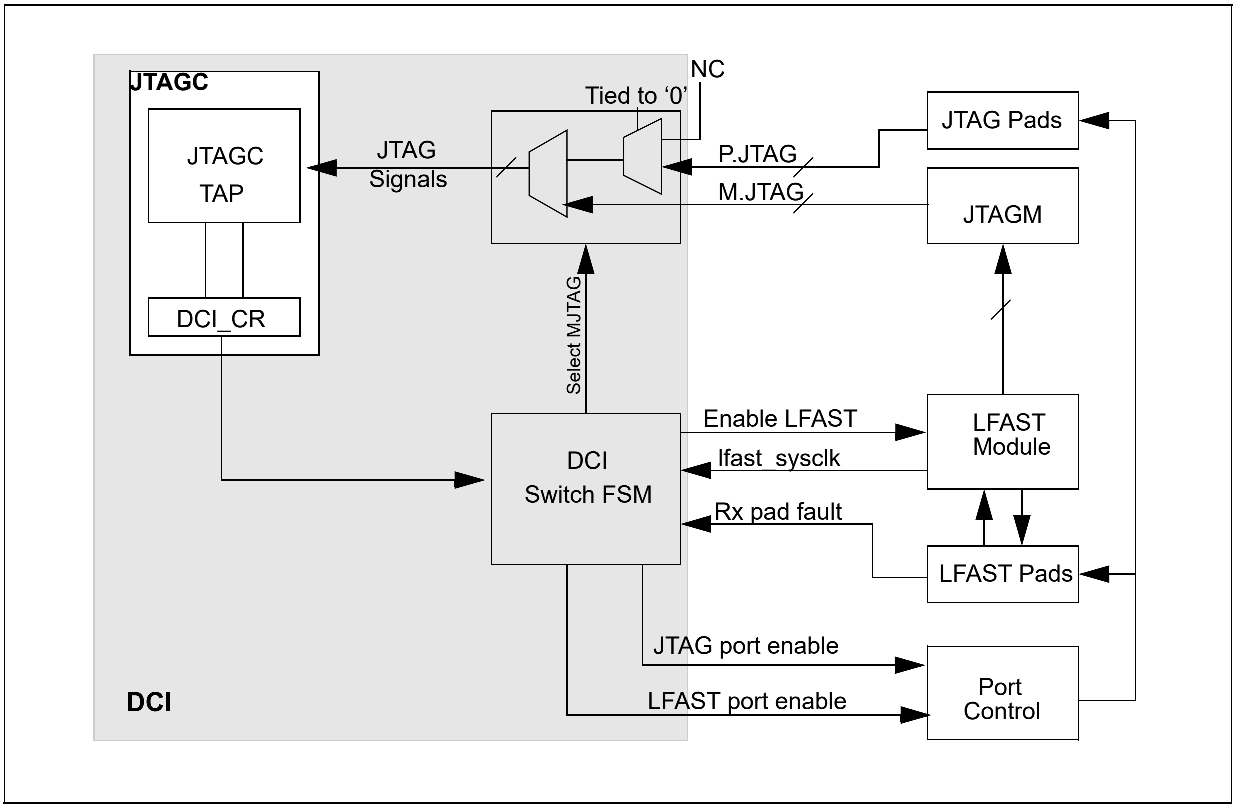
Table 1221 contains the operating mode descriptions.

Table 1221. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.
DCI_JTAG_SAFE	SAFE_JTAG state. When an external tool sets the DCI_CR[EN_LFAST] bit, this indicates that the external tool is going to switch to the LFAST mode. The external tool puts the DCI in Run-Test-Idle state to start the switching operation. To enable safe switching operation, the DCI enters the JTAG_SAFE state where it performs the following operations: – JCOMP is latched inside to control until switching gets completed – The DCI releases control of the JTAG ports and uses high on TMS inside, thus keeping the DCI in the Run-Test-Idle state – An internal counter starts to count to a tool specified value as programmed in DCI_CR[SWITCH_CNTR1].
DCI_LFAST_SWITCH	LFAST port switching state. In this state, the DCI asserts the enable signal to indicate to the LFAST that it is now ready for LFAST port switching. In this mode, the LFAST Rx pads are enabled. The switch counter starts another count operation to count to the DCI_CR[SWITCH_CNTR2] value.
DCI_LFAST	LFAST mode with escape mode enabled. This mode indicates that the switching is now completed and the debug tool is operating in the LFAST mode. However, there is another feature called Escape Mode, which enables the LFAST pads to detect any disconnection fault on its Rx pads. When these pads are working in Escape Mode, a sensing logic is enabled in the pad which detects and flags any out of common mode voltage range for the Rx pads. If this flag is asserted, it is used to force the DCI to go to LFAST_SAFE mode to ensure safe switching to the JTAG state.
DCI_LFAST_SAFE	LFAST_SAFE mode. While the tool is switching from LFAST mode to JTAG mode, it resets the DCI_CR[EN_LFAST] bit. The external tool puts the DCI in the Run-Test-Idle state to start the switching operation. To account for the switching time for the pads, the DCI enters in a safe LFAST state. It then restarts a counter to count to a tool specified value in DCI_CR[SWITCH_CNTR1], and disables both the LFAST function on the pads and the LFAST module.

Figure 1210 shows a block diagram of the JTAG/LFAST connections.

Figure 1210. DCI JTAG/LFAST switch integration



58.1.3.1 Port sharing between JTAG and LFAST modes

By default the DCI is configured in JTAG mode using the standard JTAG port (P.JTAG). Therefore, the initial configuration is done using low speed JTAG signals. In order to switch from JTAG mode to LFAST mode, the tool accesses the following fields in the DCI control register (DCI_CR):

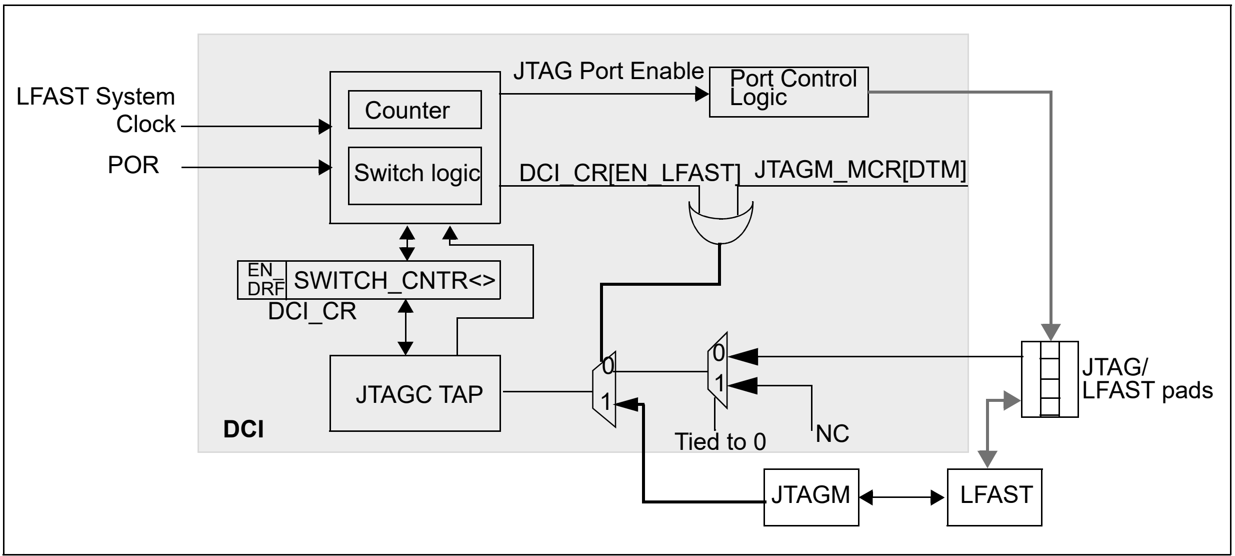
- EN_LFAST – Bit to indicate enable/disable LFAST mode
- SWITCH_CNTR1 and SWITCH_CNTR2 – Bit fields to program a count value that corresponds to the delay for switching the pin functions when changing modes.

As the JTAG to LFAST mode switching reflects change in port input/output behavior, it is done through a controlled procedure.The DCI_CR is only accessible through JTAG sequences. While switching from JTAG to LFAST mode, the tool programs this register via P.JTAG to enable switching to LFAST mode. As the pins are shared between JTAG and LFAST functions, a programmable delay based on switch counters is required which allows a safe switching for the MCU as well as for the tool to change its mode. In LFAST mode, the tool can write this register to switch back from high speed debug to low speed debug. Again, the similar switching delay mechanism is used. The switching schematic is shown in *Figure 1211*.

A programmable switching delay and tri-stating port pins during switching allow a connected tool to safely tune the device turn-around of pin functions from JTAG mode to LFAST mode and vice versa. These delays are programmed according to the time required by the tool to change its operation from JTAG mode to LFAST and vice versa. This reduces the possibility of device damage during switching.

The programmable counter uses the LFAST system clock as the clock source.

Figure 1211. DCI switching between JTAG and LFAST modes



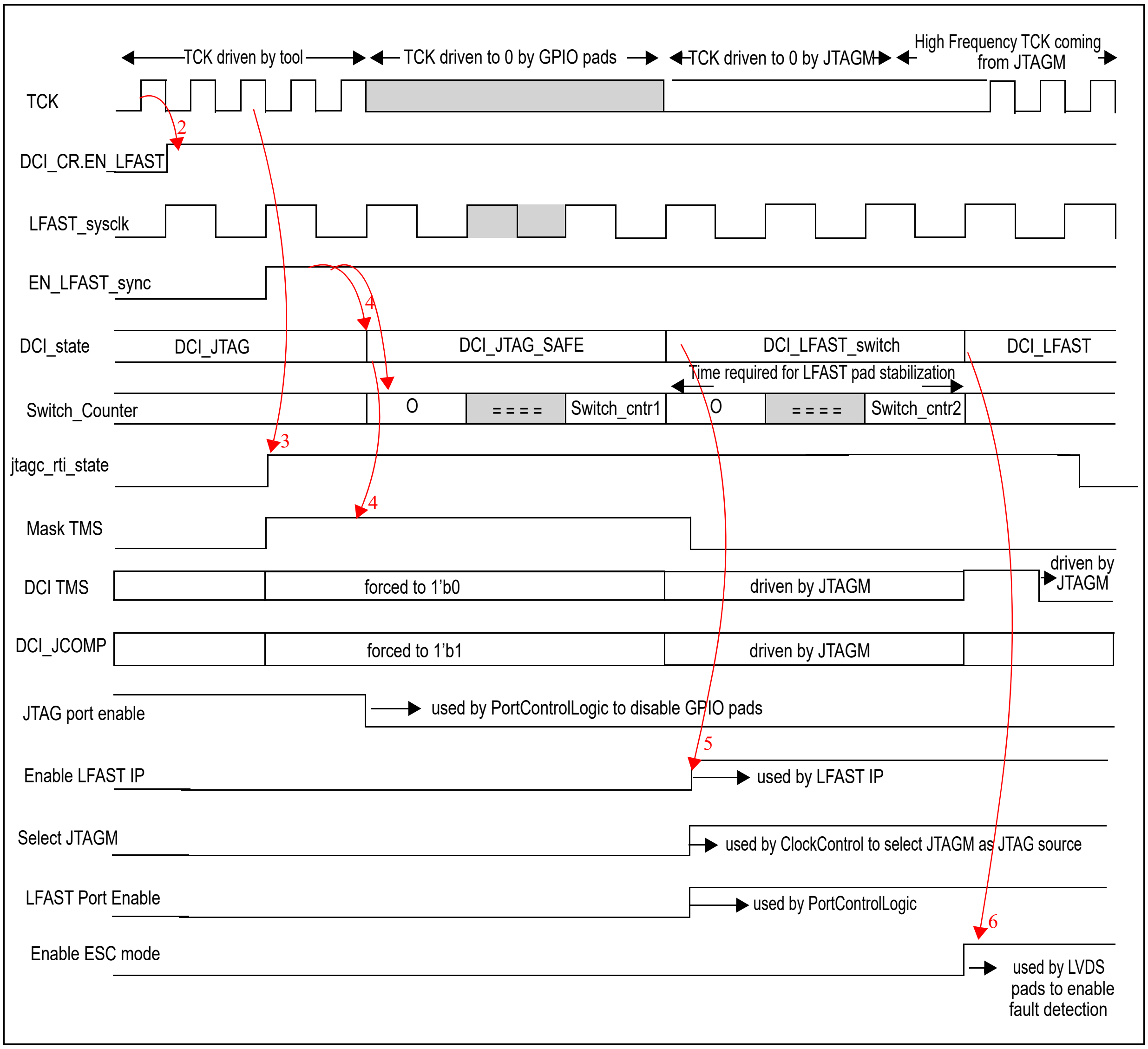
58.1.3.1.1 Switching from JTAG mode to LFAST mode

The steps to switch operation from JTAG to LFAST mode are as follows:

1. System starts in port JTAG mode after power-on-reset.
2. External tool programs the following bits in the DCI_CR:
   - EN_LFAST
   - SWITCH_CNTR1
   - SWITCH_CNTR2
3. Tool puts JTAG TAP in the Run-Test-Idle mode.
4. An internal finite state machine (FSM) starts the switching operation using the LFAST reference clock. The switching process takes four LFAST reference clocks to start, and an external tool should not change the JTAG signals during this period. The DCI forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. The DCI also triggers a switching counter to allow for some time for the tool to change its mode from JTAG to LFAST.
5. After the counter counts to the DCI_CR[SWITCH_CNTR1] value, the LFAST IP and LFAST pads are enabled. Then the counter is reset. Also, internally JTAGM is selected as the JTAG source for the DCI.
6. Counter counts to the DCI_CR[SWITCH_CNTR2] value to enable the Escape mode feature for the LFAST Rx pads. This time allows the tool to settle down in LFAST mode.
7. Tool starts LFAST transmission when it detects pulses on the DRCLK pin. Tool now can use the LFAST to program the JTAGM to generate JTAG sequences.

Figure 1212 shows the timing sequence.

Figure 1212. DCI JTAG to LFAST switching diagram



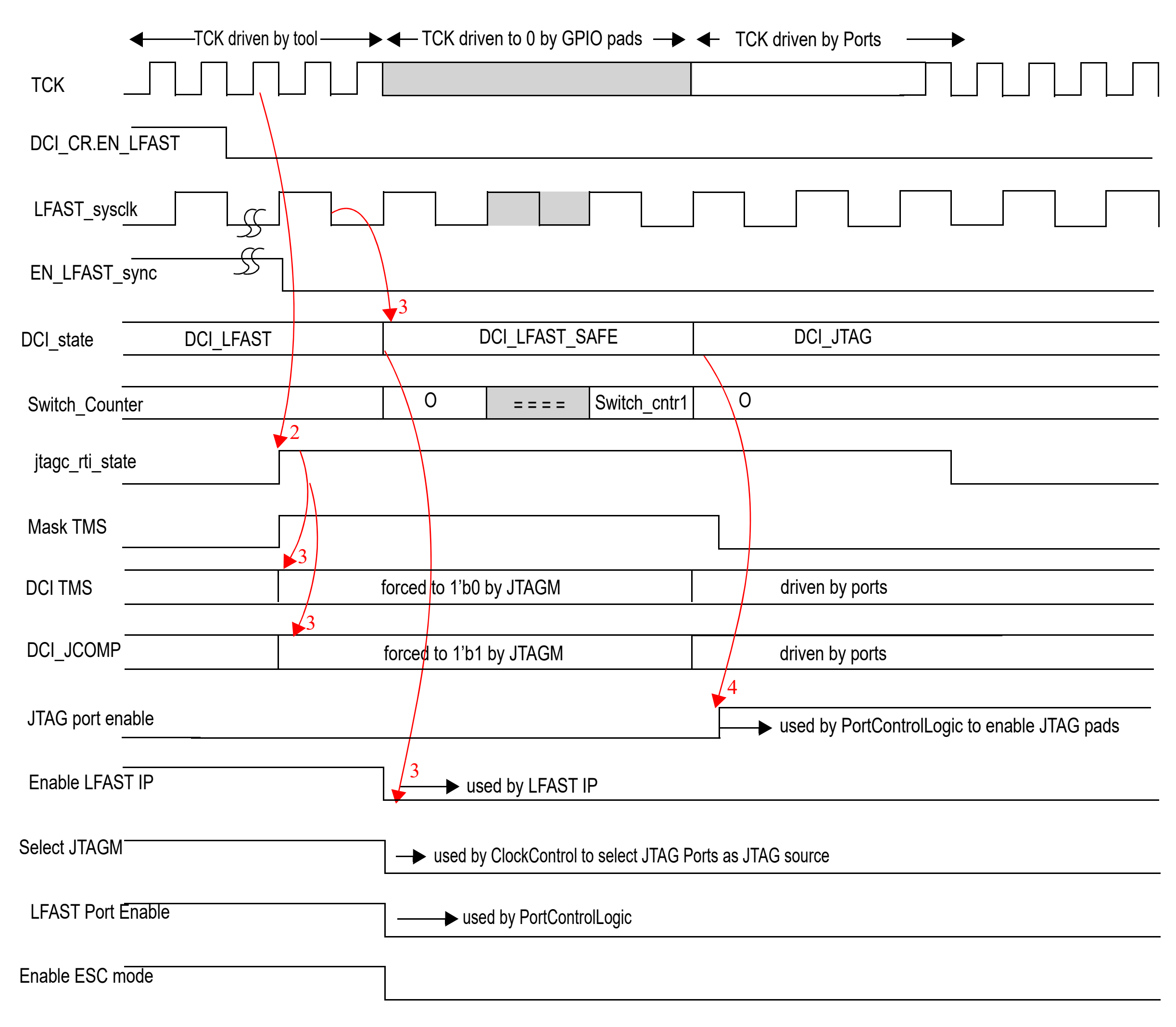
58.1.3.1.2 Switching from LFAST mode to JTAG mode

The steps to switch operation from LFAST to JTAG mode are as follows:

1. External tool programs the DCI_CR to reset the EN_LFAST bit and SWITCH_CNTR1 for the switching delay.
2. Tool puts JTAG TAP in the Run-Test-Idle mode.
3. An internal FSM starts switching operation. It forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. It also triggers a switching counter to count to DCI_CR[SWITCH_CNTR1] to allow for the tool to change its mode from LFAST to JTAG.
4. After the counter overflows, the JTAG pads are enabled and JTAGM releases its control for the DCI JTAG source.

Table 1225 and *Table 1227* summarize the MCU and tool behaviors. *Figure 1213* shows the timing sequence.

Figure 1213. DCI LFAST to JTAG switching diagram



58.1.3.2 VSS detection

When the LFAST pads are operating in Escape mode, a sensing logic is enabled in the pad which detects and flags any disconnection of the LFAST Rx pads. This flag, if asserted, is used to force the DCI to go to JTAG mode to ensure safe operation. This flag, when asserted, also resets the JTAG TAP to restart in JTAG mode and the EN_LFAST bit is reset accordingly. This flag resets only JTAG TAPs and does not affect any Nexus blocks.

This feature is enabled only when DCI_CR[EN_ESC_MODE] is set. By default this feature is not enabled. To tune the tool behavior it is possible to emulate the pad fault using the DCI_CR[FLVDS_PAD_FLT] bit.

58.1.3.3 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool or LFAST-based tool, it is possible to generate JTAG sequences through software programming using JTAGM IP. The JTAGM, which generates JTAG sequences using LFAST protocol, is also used to generate JTAG sequences using software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

When MCR[jtagm_JCOMP] is programmed to 0, the JTAGC block reset is asserted, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

Note: It is recommended to utilize either hardware based debug (JTAG or LFAST) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

If both debug methods are used simultaneously, then an external tool may arbitrarily connect when the SoC has already been halted using the software debug mode. In this case, two consecutive assertions of external JCOMP are necessary to allow the external tool to take control of debug session.

58.1.3.4 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect
- JCOMP
- SoC Debug_Enable signal
- System PLL is out of lock state and not bypassed

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the Nexus reset signal remains asserted until the system clock achieves lock or in bypass mode, as indicated by the system clock lock status signal, and Debug_Enable is asserted and JCOMP is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.If Debug_Enable is de-asserted by the SoC security modules, then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control, and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 1222. Nexus reset truth table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

58.1.3.5 SWT control

The DCI is also used to enable or disable the Software Watchdog Timer (SWT). The DCI provides a single bit SWT enable signal which is used to control all SWTs in the MCU. For this, the $\overline{\text{EVTI}}$[0] pin is used. If the $\overline{\text{EVTI}}$[0] pin remains high when the tool asserts JCOMP, then the SWT is enabled. But if the tool drives the $\overline{\text{EVTI}}$[0] pin low before asserting JCOMP pin, then the SWT is disabled. By default the SWT is enabled and a tool has an option to enable/disable the SWT using the $\overline{\text{EVTI}}$[0] pin.

Note: Debugger should check the SWT counter to confirm if SWT has actually stopped or not, otherwise repeat the sequence mentioned above to disable the SWT.

58.1.3.6 Break control

The DCI provides central break control among the CPUs and non-CPU debug modules such as DMA, Watchdog and other peripherals.

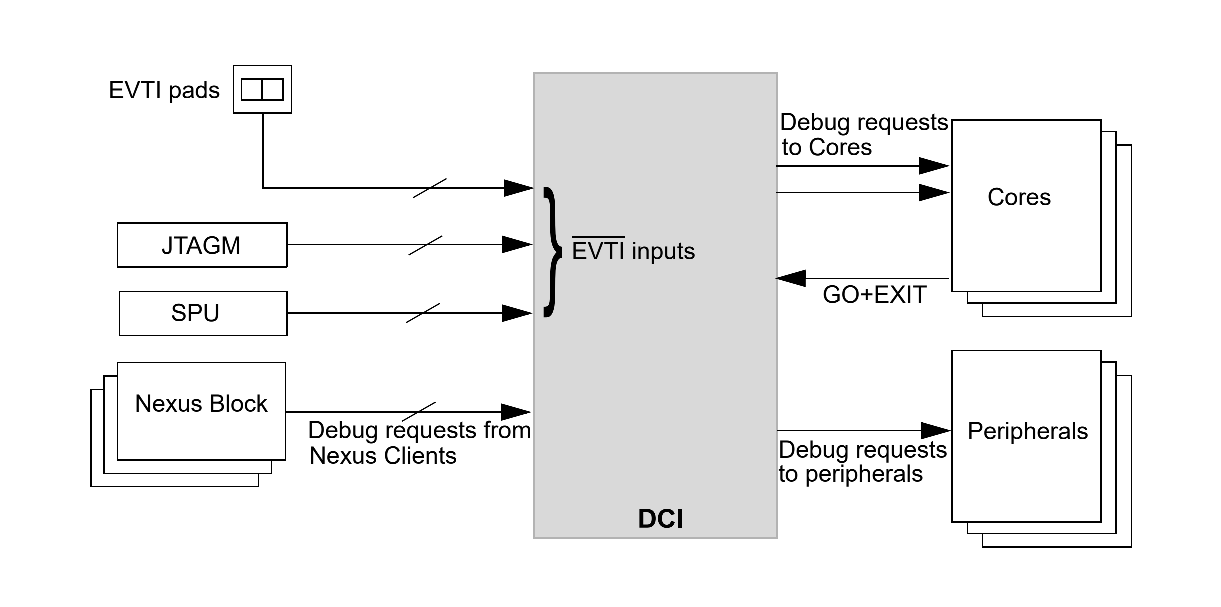
There are various different input sources that provide debug requests:

- $\overline{\text{EVTI}}$[0] signal from EVTI pad
- $\overline{\text{EVTI}}$[1:0] signals from JTAGM
- $\overline{\text{EVTI}}$[1:0] signals from the Sequence Processing Unit (SPU)

The DCI generates following debug request signals:

- Debug Request signal to be used for all non-CPU peripherals
- External Debug Request to be used to request external debug event to processor cores
- OnCE Debug Request to be used to request OnCE debug event to processor cores
- All break control requests are issued simultaneously to CPUs and peripherals, then synchronized to each clock domain

Figure 1214. DCI debug control mechanism



The various sources of EVTI inputs are synchronized and logically ANDed in the DCI to generate the following:

- debug requests for each processor core
- external debug requests for each processor core
- debug request for all non-CPU peripherals

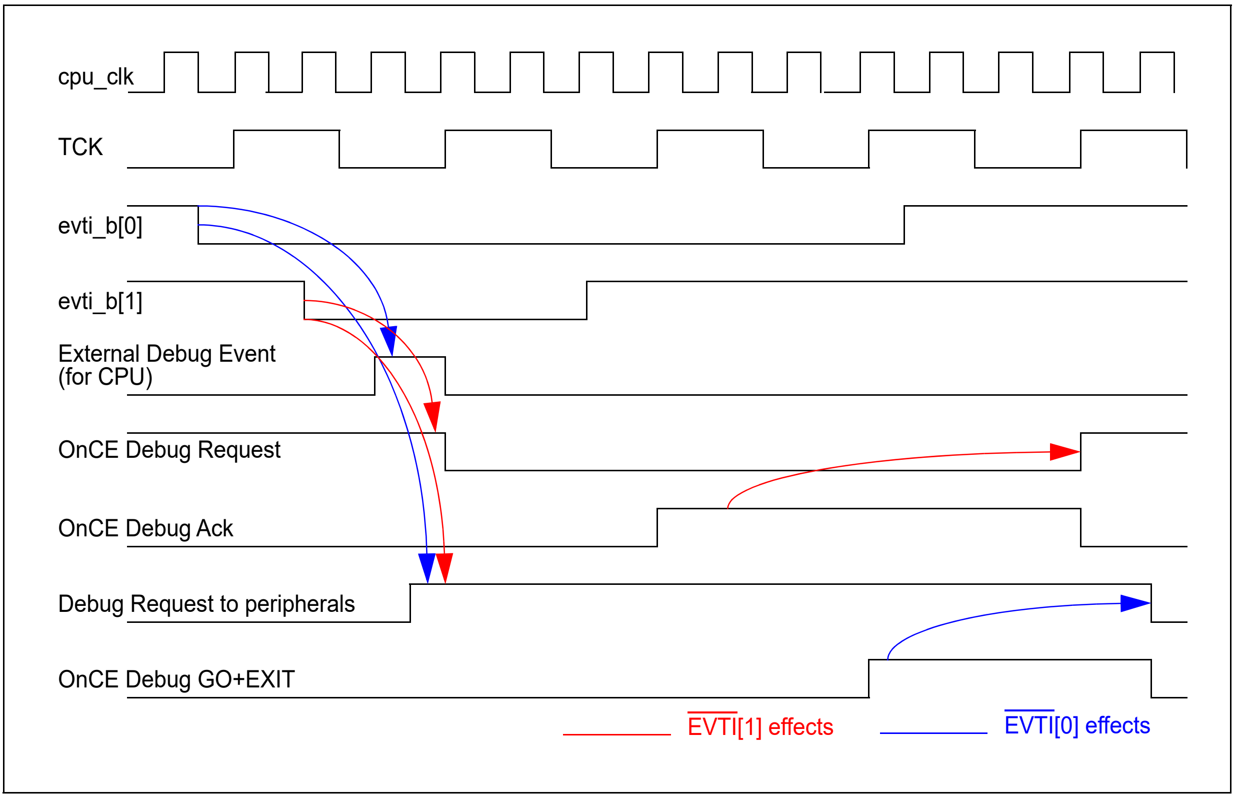
The EVTI[0] from the EVTI pads, SPU and JTAGM are used to generate debug requests for non-CPU peripherals and for processor cores through the external debug request signal. The EVTI[1] from the SPU and JTAGM, is also used to generate debug requests for non-CPU peripherals and it generates debug requests to processor cores through the OnCE debug request signal.

The external debug request to processor core is asserted as a single cycle pulse and is synchronized on the respective core clock. The OnCE debug request is negated when the acknowledge signal from the Cores is asserted.The CPU(s) remain in the debug state until GO+EX command is executed for that CPU. Exiting a CPU out of debug state by executing GO+EX command does not take other CPUs out of debug state. The peripheral debug request is negated when the processor core(s) execute GO+EXIT command.The EVTI[0] signal coming from the pad is enabled for debug control using DCI_CR[4] bit.

In order to selectively enable/disable debug requests to system peripherals, the respective debug enable bit inside the peripheral should be used.

Figure 1215 shows the timing sequence.

Figure 1215. DCI debug signal generation



58.1.3.7 Cross-triggering control

Many of the blocks in the device, including the Nexus trace blocks and more sophisticated peripherals, have debug features that allow them to request device-wide breaks to enter debug mode. Many blocks can be programmed to halt operation upon debug mode entry or ignore it and continue normal operation. All debug mode entry requests are input to the DCI, ORed together and the result is used to produce the debug requests to the CPUs and non-CPU peripherals.

The cross-triggering feature is implemented through selectively enabling debug control bits in the respective modules. Processor cores (enabled using DBCR0[DEVT1] bit) and non-core peripherals can be programmed to halt operation through their respective control bits. Cross-triggering provides a selective feature to put modules in debug mode upon receiving halt request signals from Nexus modules.

58.1.3.8 Synchronous restart control

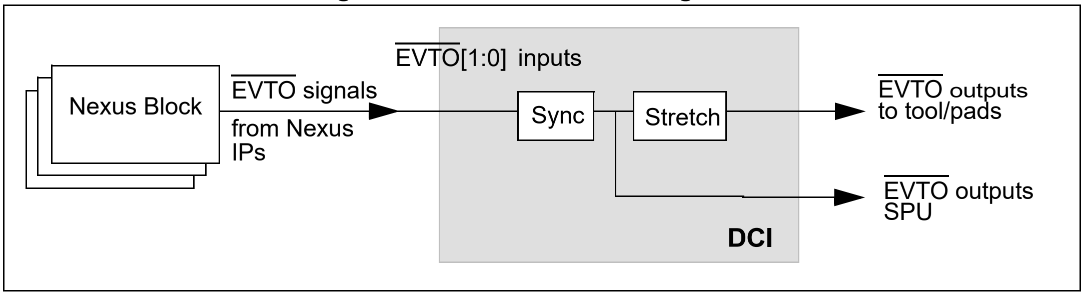
The DCI implements a synchronous restart control feature to synchronously restart all debug peripherals (CPU and non-CPU peripherals which are put in debug state). This is accomplished with the following steps:

1. The tool sequentially scans into CPUSCR for each processor core to restore the proper processor state.
2. The tool scans into the Production device tap controller the AUX_PARALLEL instruction, which then selects all processor core tap controllers in parallel.
3. To synchronously exit any combination of processor cores from debug mode, the tool simultaneously scans into all processor core tap instruction registers (OCMD) the 10-bit sequence 0x180 (that is set GO and EX, clear RS[0:6]).
4. If a processor core is out of debug mode, it sets a flag OnCE GO+EXIT.
5. When the OR of the processor core OnCE GO+EXIT signals is asserted, then all other selected non-processor cores and peripherals in debug would also be synchronously exited from debug mode. It is done by negating peripheral debug request signal.

58.1.3.9 EVTO management

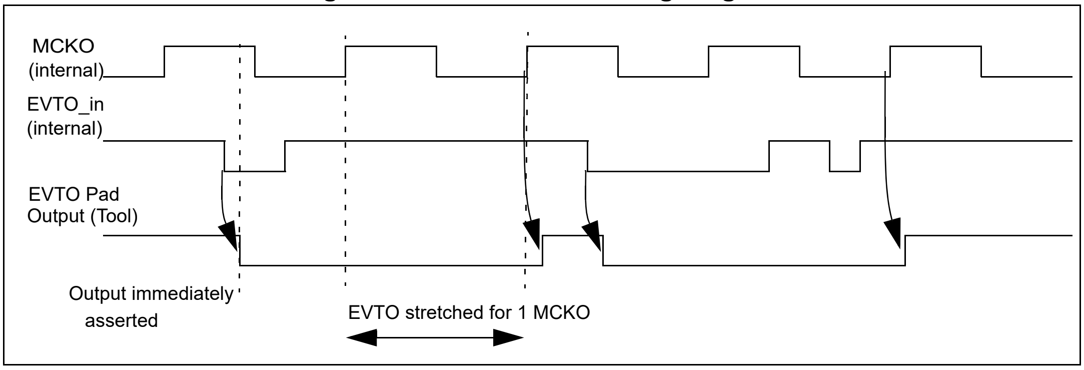
The EVTO management block diagram is presented in *Figure 1216*.

Figure 1216. DCI EVTO management



The DCI module controls distribution of the EVTO output from all Nexus Clients including the cores that produce EVTO[0] signals. The DCI uses a high frequency clock to sample EVTO signal coming from all modules which might generate EVTO outputs at their own clock frequency. After receiving a single clock period of asserted EVTO[0] signals from any Nexus client, the DCI immediately asserts EVTO[0] outputs and stretches it for a minimum of one output clock period for the output to the tool(*Figure 1217*). The DCI also generates EVTO output in other device clock domains so that other Nexus clients can receive EVTO input without the need of any synchronization. EVTO function is active as long as the DCI is not in reset.

Figure 1217. DCI EVTO timing diagram



58.1.3.10 Reset control

The DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force functional reset—This reset is generated by programming the DCI_CR[FPD_FUNC] bit. When set, it forces functional reset. This bit has to be reset by the tool to negate the reset signal.
- Force POR or destructive reset—This reset is generated by programming DCI_CR[FPD_DEST] bit. When set it forces destructive reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

58.1.3.11 Security

The SoC security logic provides a Debug_Enable signal to the DCI. The DCI uses the Debug_Enable signal as a control input to generate debug module reset. When debug disable is requested, the DCI keeps all debug blocks in the reset state. This is a protection mode used to disable debug accesses to the peripheral when the device is in a secured state.

58.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

58.3 Register description

The DCI module contains one registers, DCI_CR, which is present inside the JTAGC module. This register is only controlled using the JTAGC TAP and no memory-mapped registers are present.

58.3.1 DCI control register (DCI_CR)

The DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. *Figure 1218* shows the format of the DCI_CR.

	31	30	29	28	27	26	25	24
R W	SWITCH_CNTR2							
Reset	1	1	1	1	1	1	1	1

Figure 1218. DCI control register (DCI_CR)

	23	22	21	20	19	18	17	16
R/W	SWITCH_CNTR2							
Reset	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8
R/W	SWITCH_CNTR1					FLVDS_PAD_FLT	FPD_FUNC	FPD_DEST
Reset	1	1	1	1	1	0	0	0

	7	6	5	4	3	2	1	0
R/W	Reserved	FORCE_LFAST	Reserved	EVTI0_PAD_EN	Reserved	EVTO0_PAD_EN	EN_ESC_MODE	EN_LFAST
Reset	0	0	0	0	0	0	0	0

Figure 1218. DCI control register (DCI_CR) (continued)

The DCI_CR is described in *Table 1223*.

Table 1223. DCI_CR field descriptions

Field	Description
31:16 SWITCH_CNTR2	Width of Switch_CNTR2 Decides the second timeout value while switching from JTAG to LFAST mode. The maximum timeout duration = 32,768×T_{lfast_sysclk}
15:11 SWITCH_CNTR1	Width of Switch_CNTR1 Decides the first timeout value while switching from JTAG to LFAST mode, and the timeout value while switching from LFAST mode to JTAG mode. The maximum timeout duration = 32×T_{lfast_sysclk}
10 FLVDS_PAD_FLT	Force LVDS Pad Fault Forces the pad fault condition to tune the tool without actually having the pad fault on LFAST Rx pads. This bit is effective only when DCI_CR[EN_ESC_MODE] is set. 0 Do not force Rx LVDS Pad Fault 1 Force Rx LVDS Pad Fault if DCI_CR[EN_ESC_MODE] is set
9 FPD_FUNC	Force Functional reset 0 Do not force functional reset 1 Force functional reset

Table 1223. DCI_CR field descriptions (continued)

Field	Description
8 FPD_DEST	Force Destructive reset 0 Do not force destructive reset 1 Force destructive reset
6 FORCE_LFAST	Force enable for LFAST 0 LFAST not forced to enable and is controlled using DCI_CR[EN_LFAST] bit 1 LFAST forcefully enabled (without using DCI_CR[EN_LFAST] bit)
4 EVTI0_PAD_EN(1)	$\overline{\text{EVTI}}$[0] feature enable 0 $\overline{\text{EVTI}}$[0] disabled (default) 1 $\overline{\text{EVTI}}$[0] enabled
2 EVTO0_PAD_EN	Output enable for $\overline{\text{EVTO}}$[0] pad 0 $\overline{\text{EVTO}}$[0] pad output not enabled (default) 1 $\overline{\text{EVTO}}$[0] pad output enabled
1 EN_ESC_MODE	Enable Escape Mode Fault detect feature enable for LFAST Rx pads. 0 Fault detect feature not present for LFAST Rx pads (default) 1 Fault detect feature present for LFAST Rx pads and are enabled after the successful switching to LFAST mode
0 EN_LFAST(1)	Enable for LFAST-based tool selection This bit can be reset either by programming by tool, or pad fault condition in which JTAGC block is reset. 0 LFAST tool not enabled. DCI operates in JTAG mode (default) 1 LFAST tool enabled

1. DCI_CR is a JTAG data register and is not accessible for software read. However, the DCI_CR[EN_LFAST] bit is mapped to JTAGM register bit JTAGM_SR[1] to allow for software read access and a mechanism for software to be aware of an active debug tool interface.

58.4 Functional description

58.4.1 DCI mode transition

After power-on reset, the DCI starts in JTAG mode. To enable LFAST mode, the external tool has to first operate in JTAG mode, configure DCI and then switch to LFAST mode. This requires certain careful steps so as to avoid any contention on double-bonded pads for JTAG/LFAST. *Table 1224* and *Table 1225* show the steps and expected states for the MCU and tool for JTAG to LFAST switching. *Table 1226* and *Table 1227* contain the different steps, and the MCU and tool expected states for the LFAST to JTAG switching.

Table 1224. MCU operation (JTAG to LFAST switching)

MCU pin	JCOMP low	JCOMP high (JTAG mode)	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	1st Timeout	2nd Timeout	LFAST ICLC enable command received
TDI	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxN
TMS	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxP
TDO	Hi-Z	Logic Output	Logic Output	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxP
JCOMP	Logic Input	Logic Input	Logic Input Latch	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxN
TCK	Logic Input	Logic Input	Logic Input	Hi-Z	Logic Output	Logic Output	Logic Output
Mode Escape	Inhibit	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable
LFAST_ SYSCL K _EN	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable	Enable

Table 1225. Tool operation (JTAG to LFAST switching)

Tool	JCOMP low	JCOMP high (JTAG mode)(1)	DCI_CR [EN_ LFAST]=1	Run-Test-Idle state	TCK detected	Internal timer elapse
TDI	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxN
TMS	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxP
TDO	Hi-Z	Logic Input	Hi-Z	Hi-Z	LVDS Tx	LVDS RxP
JCOMP	Hi-Z	Logic Output (High)	Hi-Z	Hi-Z	LVDS Tx	LVDS RxN
TCK	Don't Care	Logic Output	Logic Input	Logic Input	Logic Input	Logic Input

1. This state is used to set DCI_CR[EN_LFAST] bit.

Table 1226. MCU operation (LFAST to JTAG switching)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
TDI	LVDS Tx	Logic Input	Logic Input
TMS	LVDS Tx	Logic Input	Logic Input
TDO	LVDS Rx	Hi-Z	Logic Output
JCOMP	LVDS Rx	Logic Input Latch	Logic Input (sample JCOMP pin)
TCK	Logic Output	Logic Input	Logic Input

Table 1226. MCU operation (LFAST to JTAG switching) (continued)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
Mode Escape	Enable	Inhibit	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit	Inhibit

Table 1227. Tool operation (LFAST to JTAG switching)

Tool	Send command to clear DCI_CR[EN_LFAST]	LFAST transmits idle after command sent	TCK input clock stops or LVDS Rx both low	Internal timeout: start sending JTAG messages
TDI	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TMS	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TDO	LVDS Tx	Hi-Z	Logic Input	Logic Input
JCOMP	LVDS Tx	Hi-Z	Logic Output (High)	Logic Output (High)
TCK	Logic Input	Logic Input	Logic Output	Logic Output

58.4.2 LFAST LVDS pad fault

It is possible that when the tool is operating in LFAST mode, the tool may get disconnected due to external disturbances, which could put the MCU in an undetermined state. It is imperative to keep the MCU in a safe state. Thus the LFAST Rx pads have a sensing logic which detects if the Rx pad voltages are not equal to the common mode voltage when the DCI is in LFAST mode. If it is found that the LFAST Rx pads are not corresponding to the common mode voltage, the pad logic sends a signal to the DCI which then puts the DCI into JTAG mode. *Table 1228* and *Table 1229* contain the different steps and expected states for the MCU and tool for LFAST to JTAGRST switching.

Table 1228. MCU operation (LFAST to JTAGRST switching)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
TDI	LVDS TxN	Logic Input
TMS	LVDS TxP	Logic Input
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Input
TCK	Logic Output	Logic Input
Mode Escape	Enable	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit
Internal TRST	Negated	Asserted

Table 1229. Tool operation (LFAST to JTAGRST switching)

Tool	LFAST mode	Tool forces JCOMP low
TDI	LVDS TxN	Don’t Care
TMS	LVDS TxP	Don’t Care
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Output (Low)
TCK	Logic Input	Don’t Care

59 JTAG Controller (JTAGC)

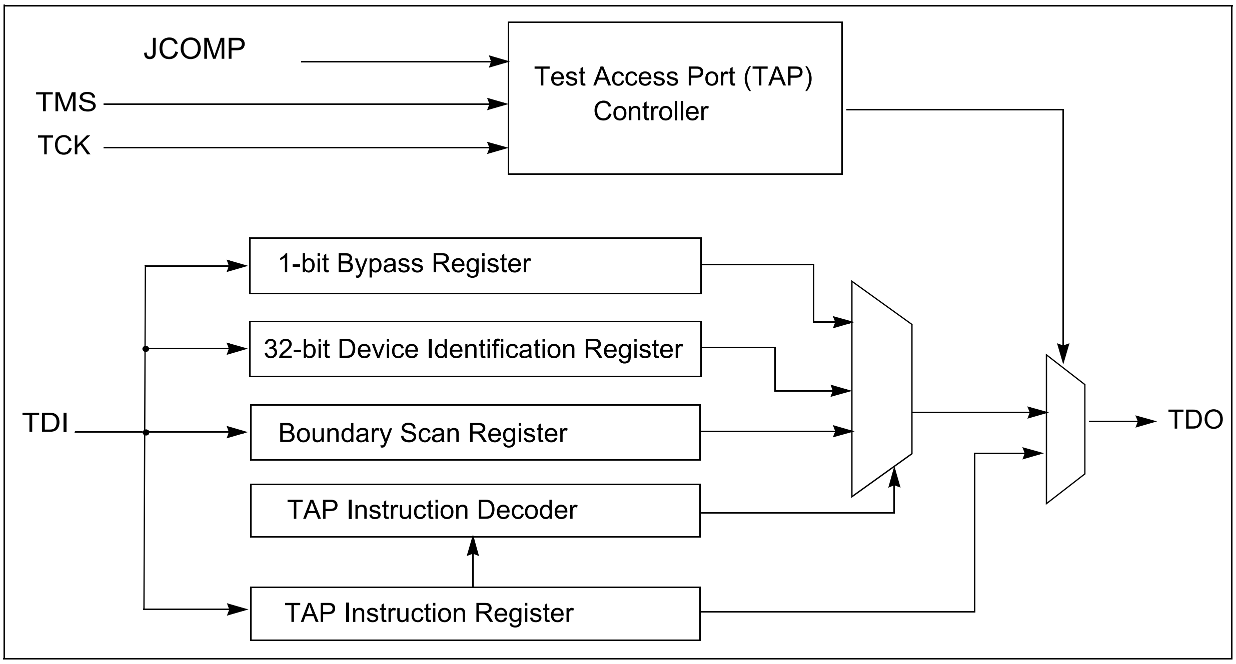
59.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

59.1.1 Block Diagram

The following is a block diagram of the JTAG Controller (JTAGC) block.

Figure 1219. JTAG (IEEE 1149.1) block diagram



59.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control and the ability to share the TAP.
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to *Table 1233* for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Data registers, bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

59.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

59.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered.
- The instruction register is loaded with the IDCODE instruction.

59.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in *Section 59.4.4*.

59.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

59.2 External signal description

The JTAGC consists of five signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 1230. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z(1)	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

59.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

59.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

59.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in *Section 59.4.3*.

59.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

59.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

59.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

59.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	0	1	2	3	4	5
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

Figure 1220. 6-bit instruction register

59.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

59.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Minor Mask				Major Mask				Design Center		Technology Node		Application Family			Sub Family[0]
W																
Reset	0	0	0	0	0	0	0	0	0	0	-[1]	-[1]	-[1]	-[1]	-[1]	-[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Sub Family[1:3]			0	Manufacturer ID											0
W																
Reset	-[1]	-[1]	-[1]	0	0	0	0	0	0	1	0	0	0	0	0	1

1. The value is shown in *Table 1231*

Figure 1221. Device identification register

The following table describes the device identification register functions.

Table 1231. Device identification register field descriptions

Field	Description
0:3 Minor Mask	Minor Mask These bits contain the minor mask of the part
4:7 Major Mask	Major Mask These bits contain the major mask of the part
8:9 Design Center	Design Center These bits indicate the device design center
10:11 Technology Node	Technology Node These bits indicate the device technology node 01 40nm
12:14 Application Family	Application Family These bits indicate the device application family 010 SPC58 - Body 40nm
15:18 Sub Family	Sub Family These bits indicate the device sub family 0010 SPC584B70
20:30 Manufacturer ID	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID 00000100000: STMicroelectronics

59.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256’b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.

	0	1	2	...	256
R W	JTAG_PASSWORD				
Reset:	*(1)	*	*	*	*

1. The reset value of JTAG_PASSWORD is 256'b0.

Figure 1222. JTAG_PASSWORD register

The following table describes the JTAG_PASSWORD register functions.

Table 1232. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

59.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in *Section 59.4.5*. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

59.4 Functional description

This section explains the JTAGC functional description.

59.4.1 JTAGC reset configuration

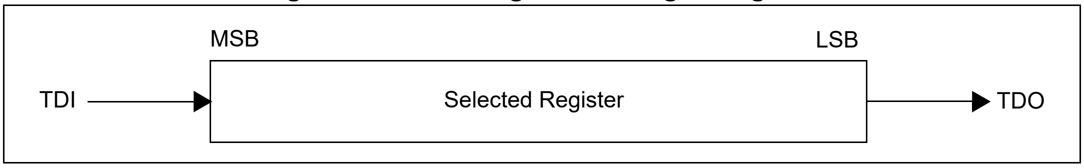
While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

59.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to *Section 59.4.4.8: ACCESS_AUX_x instructions*.

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

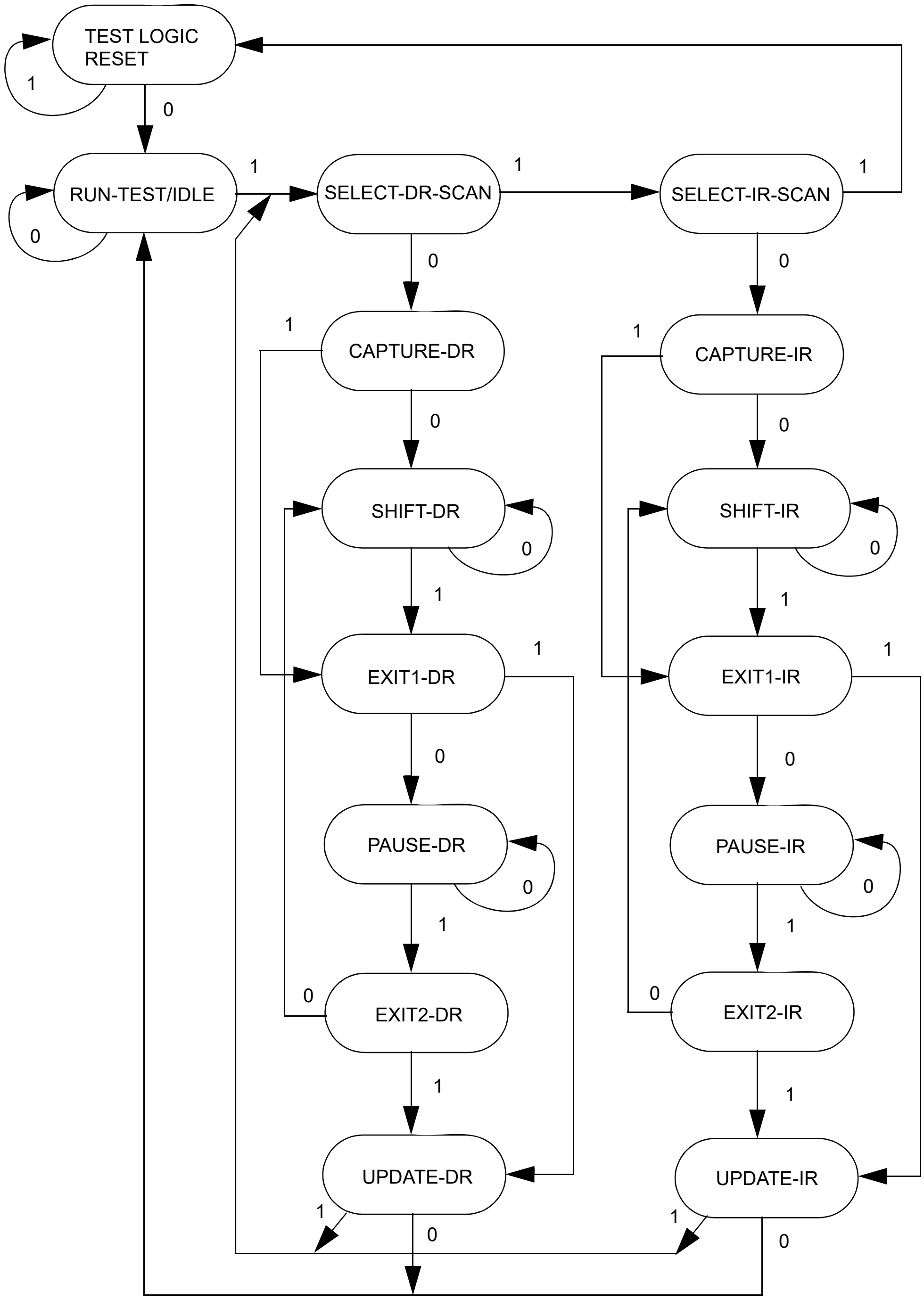
Figure 1223. Shifting data through a register



59.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine’s states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.

Figure 1224. IEEE 1149.1-2001 TAP controller finite state machine



Note:

1. The value shown adjacent to each state transition represents the value of TMS at the time of a rising edge of TCK.

59.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

59.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

59.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 1233. 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
Reserved	000101	Reserved
Reserved	000110	Reserved
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register while three-stating all output pins and asserting functional reset
Reserved	001010	Reserved
CLAMP	001100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ENABLE_DCI_CR	001110	Enables access to DCI CR register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
Reserved	100000	Reserved
ACCESS_AUX_NPC_PD	100001	Enables access to the NPC TAP controller on production device
Reserved	100001	Reserved

Table 1233. 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	100010	Reserved
Reserved	100011	Reserved
Reserved	100100	Reserved
Reserved	100101	Reserved
ACCESS_AUX_JDC	100110	Enables access to the JDC TAP controller, to access the HSM challenge and response registers
ACCESS_AUX_HSM	100111	Enables access to the HSM core TAP controller
Reserved	101001	Reserved
ACCESS_AUX_CORE_2	101010	Enables access to E200 Core 2
Reserved	101010	Reserved
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
Reserved	110011	Reserved
ACCESS_AUX_BUS_MON_0	110100	Enables access to NXMC Client 0 TAP controller
ACCESS_AUX_BUS_MON_1	110101	Enables access to NXMC Client 1 TAP controller
ACCESS_AUX_BUS_MON_2	110110	Enables access to NXMC Client 2 TAP controller
Reserved	110111	Reserved
Reserved	111000	Reserved
Reserved	111001	Reserved
ACCESS_AUX_SPU_A	111010	Enables access to SPU Client TAP controller
Reserved	111011	Reserved
Reserved	111100	Reserved
Reserved	111101	Reserved
Reserved	111110	Reserved
BYPASS	111111	Selects bypass register for data operations
Reserved[1]	All other opcodes	Decoded to select bypass register

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

59.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

59.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- First, the SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- Secondly, the PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

59.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

59.4.4.4 EXTEST—External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

59.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

59.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (for example high impedance).

HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

59.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

59.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

59.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

59.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

59.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, must be loaded the appropriate instruction for the test or action to be performed.

60 IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

60.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

60.2 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backward compatible with IEEE P1149.1. This standard is also known as “Compact JTAG” or “CJTAG”.

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths
- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

60.2.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- Compliant
  – Compliant with the IEEE 1149.1 standard
  – Four-pin operation using the “Standard Protocol” – IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
  – More than one TAPC may be associated with the TAP (once made visible by private instructions)
- Extended
  – Compatible with the IEEE 1149.1 standard
  – Four-pin operation using the “Standard Protocol” – IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
  – Additional test/debug functions
- Advanced
  – Two-pin operation using the “Advanced Protocol” – signaling where TCK(C) and TMS(C) signals transfer control/data information
  – The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

60.2.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
  - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
  - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
  - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors.
  - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
  - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

60.2.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in *Table 1234*. Signal names that end in “C” are associated with expanded functionality (above that provided by the IEEE 1149.1 standard).

The notation used in the table is as follows:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the TCKC, TMSC, TDIC, TDOC, and nTRST pins.

Table 1234. TAP.7 signal list

Signal Name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
nTRST	Test reset	O	O	O	O	O
Mandatory signal count (minimum)		2	4	4	4	4

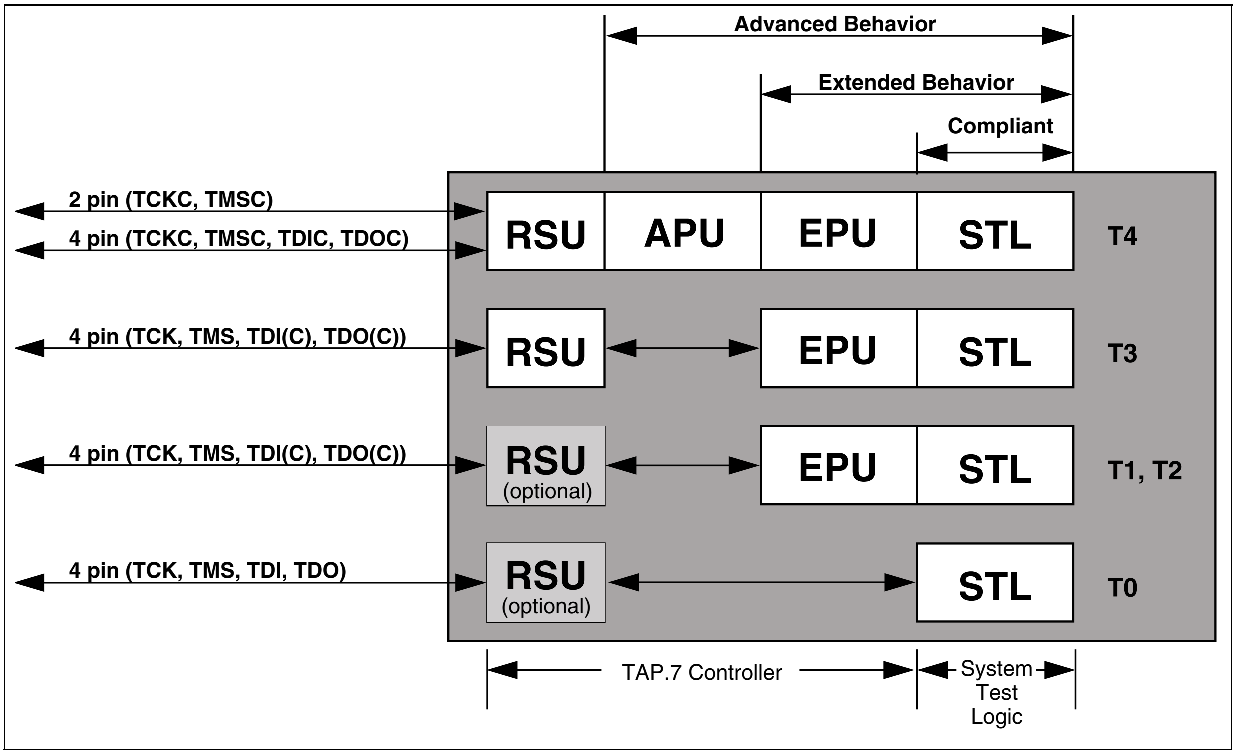
60.2.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in *Figure 1225*. It is described with the hardware layers listed below:

- **STL System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP.7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

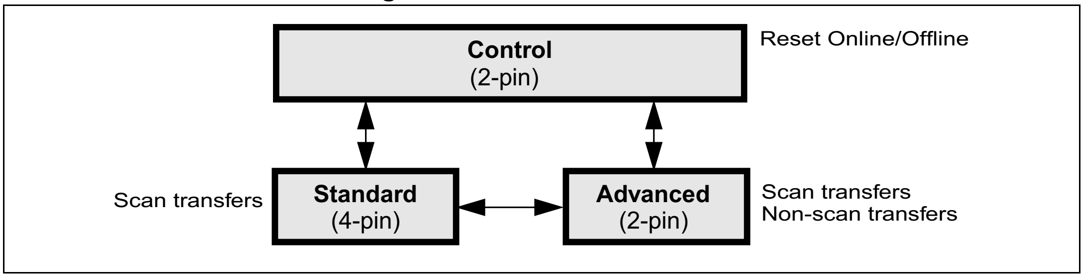
Figure 1225. TAP.7 Controller Architecture



60.2.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in *Figure 1226*.

Figure 1226. TAP.7 Behaviors



The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in *Table 1235*.

Table 1235. Class/protocol relationship

Class	Protocol		
	Standard	**Control**	**Advanced**
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

60.2.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

60.2.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMSC signaling sequences.

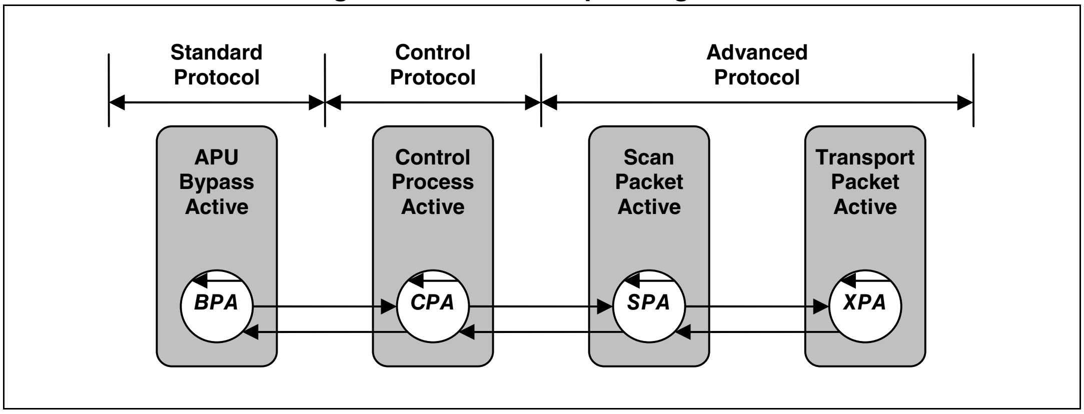
60.2.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

60.3 Operating models

Figure 1227 illustrates a simplified view of the CJTAG operating model.

Figure 1227. CJTAG operating model



60.4 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

60.4.1 T0 Functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

60.4.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

60.4.3 T2 functions

- “Super Bypass” function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

60.4.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

60.4.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

60.5 Ancillary services

60.5.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options
- Escape sequences
- TAPC state machine

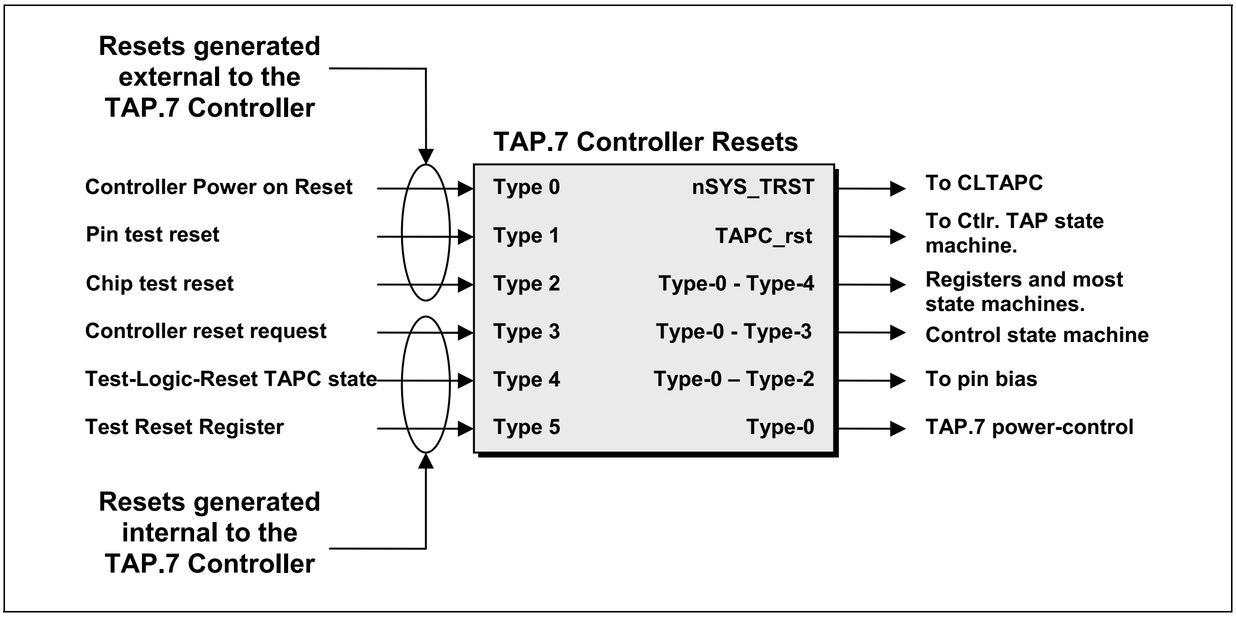
60.5.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC

Figure 1228. Conceptual view of the EPU reset logic and state machines



60.5.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

60.5.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

60.5.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

60.5.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

60.5.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

60.5.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

60.5.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

Figure 1229. Reset state machine

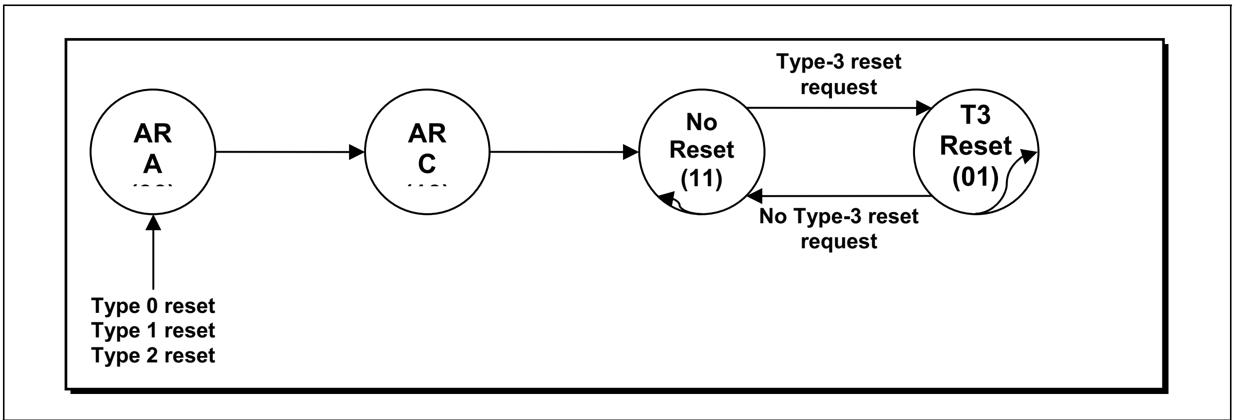


Table 1236. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in *Table 1237*.

Table 1237. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	–	–	–	–	–	–
Type-4	–	Yes	Yes	–	–	–	–
Type-3	Yes	Yes	Yes	Yes	Yes	–	–
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

60.5.3 Start-up Options

60.5.3.1 Overview

The TAP.7 start-up options are shown in *Table 1238*. The only startup option currently supported by the CJTAG module is 1149.1 compliant.

Table 1238. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to *Run-Test/Idle*

60.5.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine
- The TDIC and TDOC pins are present and provide the TDI and TDO function
- The CLTAPC is coupled at start-up

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 1239. Reset values for TAP.7 start-up options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMSC pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

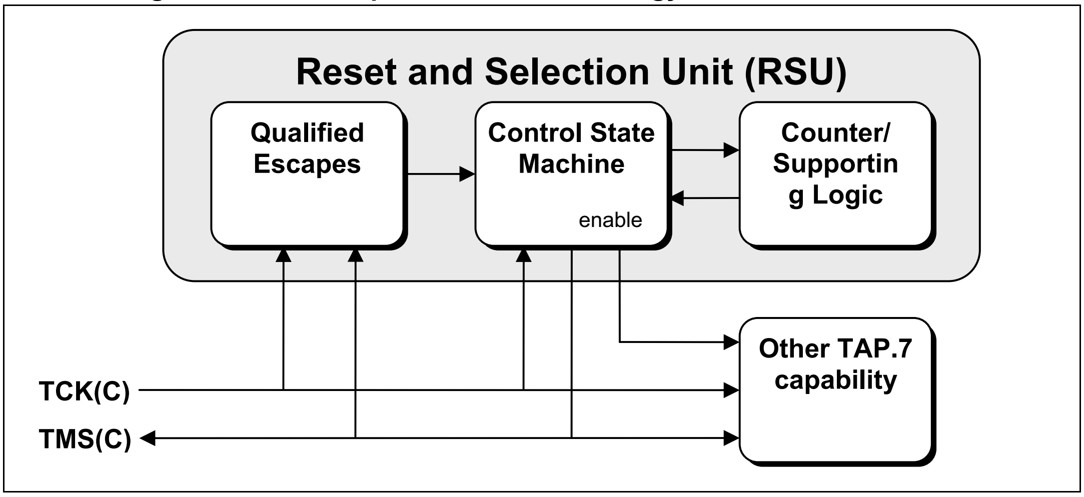
60.5.4 RSU operation

The RSU uses TCKC and TMSC with the control protocol for configuration control.

60.5.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP.7 controller is Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in *Figure 1230*.

Figure 1230. Conceptual view of technology selection mechanism



60.5.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

60.5.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges) – Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges) – Resets all technologies (generates a Type-3 TAP.7 TAPC reset).

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

60.5.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

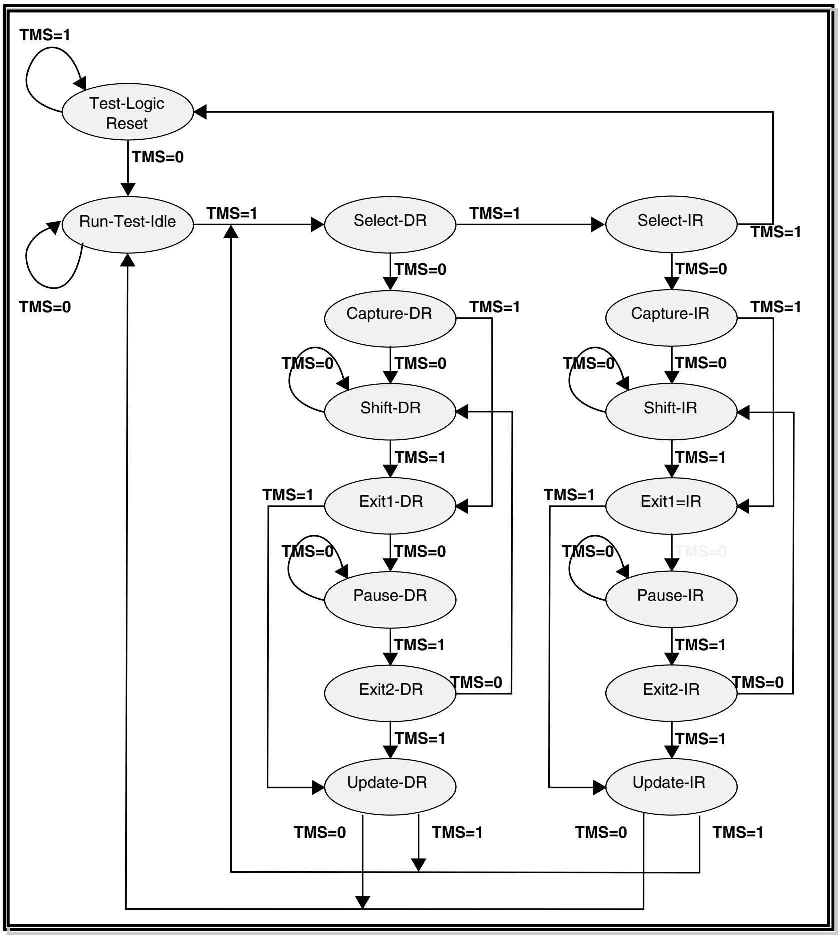
60.5.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occurs with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

60.5.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

Figure 1231. Falling edge TAP controller state machine



60.6 EPU (Extended Protocol Unit) Operation

60.6.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

60.6.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

60.6.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

60.6.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

60.6.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Accesses to optional TAP.7 scan paths
6. Force offline operation (optional)
7. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control level.

60.6.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

60.6.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

60.6.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called “***TAP.7 ZBSs***”. ZBSs that are intended for other uses are henceforth called “***Other ZBSs***”. Two TAP.7 registers enable the STL’s use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP.7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created.
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

60.6.2 EPU Registers

60.6.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose.

These commands store a single global register. Global registers manage TAP.7 TAPC functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

60.6.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers.There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back.

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- **Read-only** – Configuration and register read back.

Table 1240. TAP.7 controller register list (managed with commands)

Register type	Global/ local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
		1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Options Select	Local Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge

Table 1240. TAP.7 controller register list (managed with commands) (continued)

Register type	Global/ local	Width	Register mnemonic	Name
		32	RDBACK0	Read-back 0
		32	CNFG0	Configuration Register 0
Read-only	Local	M = Mandatory register O = Optional register		

60.6.2.3 Registers

60.6.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

60.6.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the “System Path” (STL) is used.

- **1** – Suspend use of control levels. Enables “System Path.”
- **0** – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

60.6.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

- **1** – Inhibit ZBS detection.
- **0** – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

60.6.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 1241. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	JScan0: – 1149.1 Compliant Operation. – Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	JScan1: – 1149.1 Compatible Operation – Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	JScan2: – 1149.1 Compatible Operation. – Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register
T3–T4	00011	JScan3: – Non-compatible 1149.1 Operation – Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. – Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. – Enables the use of SSDs
T4	01000	OScan0 – Data rate dependent components – Single TAP.7 communication – Debug and Test applications
	01001	OScan1 – TAP.1 components – Single/Multi-TAP.7 communication – Debug and Test applications
	01010	OScan2 – TAP.1 components – No TDI or TDO in non-shift states – Debug applications
	01011	OScan3 – TAP.1 components – No TDI in non-shift states – No TDO in any state

Table 1241. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
T4	01100	OScan4 – Data rate dependent components – Single TAP.7 communication – Debug and Test applications
	01101	OScan5 – TAP.1 components – Single/Multi-TAP.7 communication – Debug and Test applications
	01110	OScan6 – TAP.1 components – No TDI or TDO in non-shift states – Debug applications
	01111	OScan7 – TAP.1 components – No TDI in non-shift states – No TDO in any state
	10000	MScan – Data rate dependent components – Multi TAP.7 communication – Directed CID assignment – Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	Reserved

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

60.6.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

- **0** – Scan selection directives are not enabled.
- **1** – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

60.6.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance the external tool TAPC state, receive TMS and TDI information, and begin generation of a new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the “Global Register.”

60.6.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

60.6.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

- **0** – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.
- **1** – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1 value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

60.6.2.3.9 Auxiliary Pin Function Control (APFC[1:0])

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

- **0x** – TDI and TDO have JTAG function.
- **1x** – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

60.6.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

60.6.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

60.6.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0.

60.6.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as “0”.

Table 1242. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset

Table 1242. RDBACK register format (continued)

Bit	Width	Register Mnemonic	Name
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

60.6.2.3.14 Configuration Register 0 (CNFG0)

Table 1243. Configuration Register 0 Format

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class
				0000 – TAP.1 or TAP.7 Class T0
				0001 – Class T1
				0010 – Class T2
				0011 – Class T3
				0100 – Class T4
				0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision. Initial revision = all zeroes (0s)
				0000 – Reserved
				0001 – IEEE Std 1149.7-2008
				0010 – 1111 - Reserved
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format
				xxx1 – CNFG0[31:12] are implemented
				xx1x – CNFG1 implemented
				x1xx – CNFG2 implemented
				1xxx – CNFG3 implemented

Table 1243. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode x supported where x = 0,1,2
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 - Yes

Table 1243. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 – Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported
Reserved	31:29	3	Reserved	Reserved, read as zero

60.6.2.3.15 Register reset values

Table 1244. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DSGM	DSGM	DSGM	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

60.6.3 EPU Commands

There are two command types:

- Two-Part Commands
- Three-Part Commands

Any number of two-part and three-part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

60.6.3.1 Two-Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two).

The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

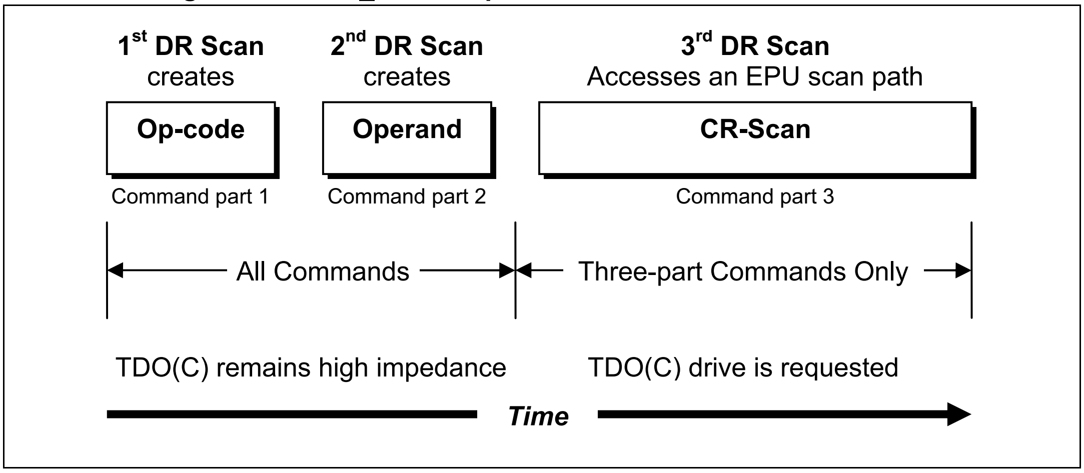
60.6.3.2 Three-Part Commands

Some commands are used to send or receive, or both, data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

60.6.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of *Shift-DR* states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in *Figure 1232*.

Figure 1232. DR_Scan sequence used for command creation



The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two-part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a ***Control-Register Scan*** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two-part commands, but only 3 three-part commands, each having a special purpose.

60.6.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in *Table 1245*.

Table 1245. TAP.7 controller command list

Command class	Op-code count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	-	M	M	M
	0x04	STDCST	Store Data Channel State	-	-	-	-
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	-	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate(1)	0x0A	CIDA	Allocate Controller ID	-	-	M	M
	0x0B	CIDD	De-allocate Controller ID	-	-	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	-	-	-	-

1. Enumerate commands are not currently supported by the CJTAG module.

60.6.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing a external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

60.6.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller’s CID is valid, and are treated as no operations otherwise.

60.6.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

60.6.3.5 Command Definitions

60.6.3.5.1 Store Commands

60.6.3.5.1.1 Store Miscellaneous Control Command (STMC)

Table 1246. Store Miscellaneous Control

STMC		Store Miscellaneous Control				
Opcode	**Operand**	**bbb**		**x**	**y**	**Description**
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	—	SGC = y
				1	—	CGM = y
		010	RdyCtl	-	—	RDYC = xy
		011	DlyCtl	-	—	DLYC = xy
		100	ScnFunc	0	—	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	—	—	Reserved
		110		—	—	
		111		—	—	

60.6.3.5.1.2 Store Conditional 1-bit (STC1)

Table 1247. Store Conditional 1 bit (STC1)

STC1		Store Conditional 1 bit			
Opcode	**Operand**	**c**	**bbb**	**CAPE**	**Description**
00001	c bbb v	0	000	—	SREDGE = v
			001	—	Reserved
			010	—	TRESET = v
			011 – 111	—	Reserved
		1	—	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

60.6.3.5.1.3 Store Conditional 2-bit (STC2)

Table 1248. Store Conditional 2 bit (STC2)

STC2		Store Conditional 2 bit			
Opcode	**Operand**	**c**	**bb**	**CAPE**	**Description**
00010	c bb vv	0	00	—	Reserved
			01	—	STCKDC = vv
			10	—	APFC = vv
			11	—	Reserved
		1	—	0	No change
			00	1	Reserved
			01	1	STCKDC = vv
			10	1	APFC = vv
			11	1	Reserved

60.6.3.5.1.4 Store Scan Format (STFMT)

Table 1249. Store Scan Format (STFMT)

STFMT		Store Scan Format
Opcode	**Operand**	
00011	nnnnn	SCNFMT = nnnnn

60.6.3.5.2 Select Commands

60.6.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 1250. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	**Operand**	**m**	**Description**
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

60.6.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 1251. Make Scan Group Candidate (MSC)

MSC		Make Scan Group Candidate	
Opcode	**Operand**	**m**	**Description**
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

60.6.3.5.3 Scan Commands

60.6.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 1252. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01000	yyyyy	-	0	Scan Group Candidate (SGC)	Write only
		-	1	Conditional Group Member (CGM)	Write only
		-	2	CNFG[0]	Read only
		-	3	CNFG[1] (not implemented)	Read only
		-	4	CNFG[2] (not implemented)	Read only
		-	5	CNFG[3] (not implemented)	Read only
		-	6	Functional reset requested (FRESETR)	Read only
		-	7	Series topology detect	Write only
		-	15 – 8	Reserved	Read only

60.6.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

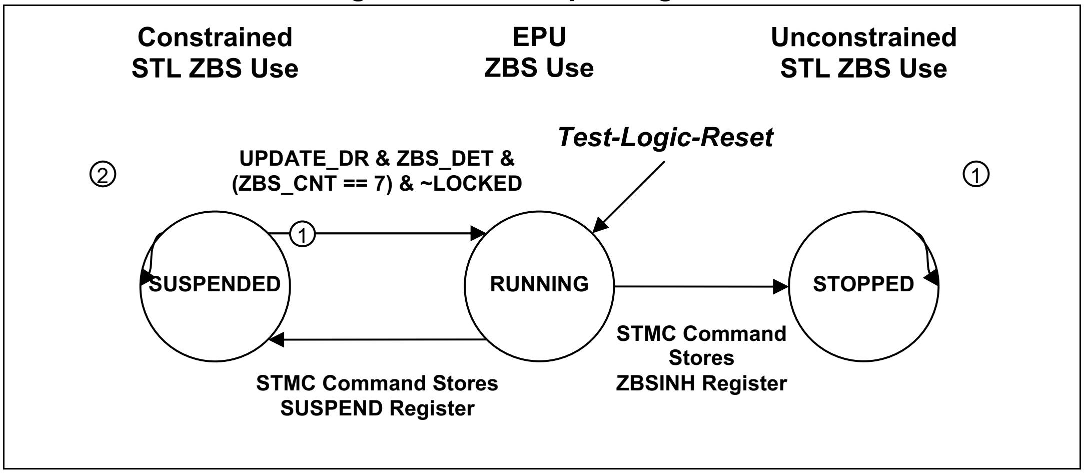
Table 1253. Scan String (SCNS)

SCNS			Scan String		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01001	yyyyy	-	0	RDBACK[0] Register [31:0]	Read only
		-	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		-	2	CNFG[0] Register [31:0]	Read only
		-	3	CNFG[1] Register [31:0] (not implemented)	Read only
		-	4	CNFG[2] Register [31:0] (not implemented)	Read only
		-	5	CNFG[3] Register [31:0] (not implemented)	Read only
		-	6 - 7	Reserved (read logic 0)	Read only
		-	31 - 8	Private – available for customization	

60.6.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in *Figure 1233*.

Figure 1233. EPU operating states



The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in *Table 1254*. The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 1254. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating state	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

60.6.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path.
- **Bit** – Supports set and test bit operations (a 1-bit shift register).
- **String** – Support 32-bit reads or 32-bit writes.
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s.
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

60.7 APU (Advanced Protocol Unit) Operation

60.7.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMSC signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (that is, ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine state progression. The EPU is unaware of the APU’s existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Three groups of scan formats (MScan, OScan0-7, and SScan0-3) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan and SScan Scan Formats are broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called “offline”. A TAP.7 controller that is not offline is henceforth called “online”. While offline the TAP.7 controller awaits the detection of a special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

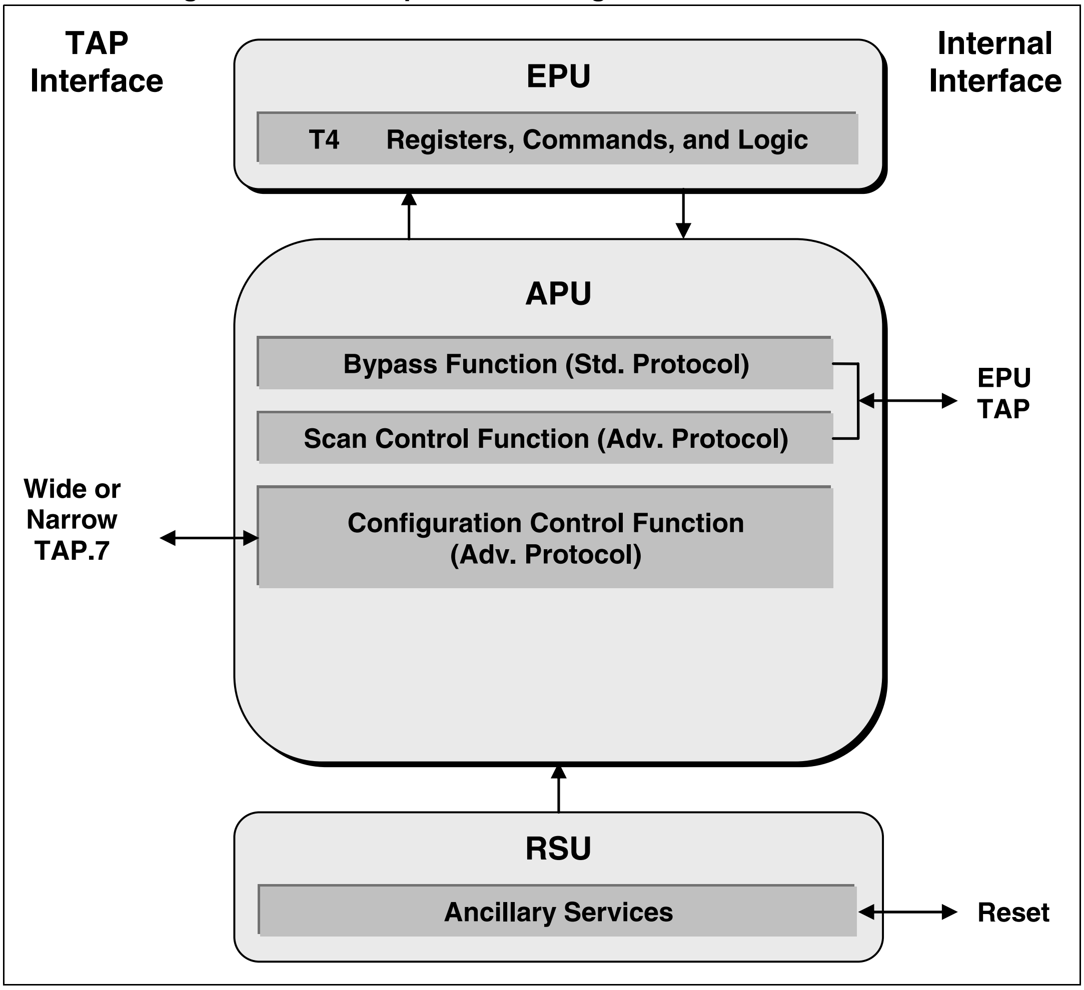
The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

Figure 1234. Conceptual block diagram of the APU functions



The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

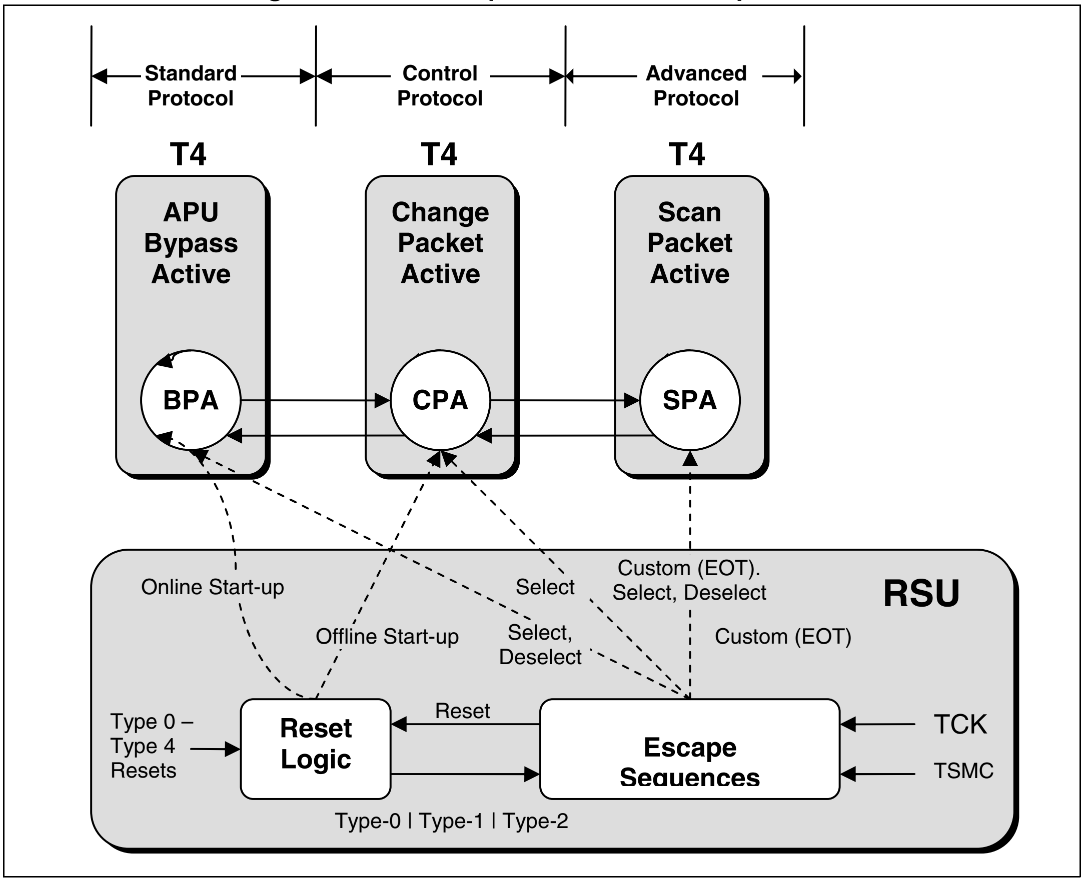
When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure (ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

60.7.2 Operation

The conceptual view of the APU operation is shown in *Figure 1235*.

Figure 1235. Conceptual view of APU operation



The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMSC pin.

APU operating states:

- ***BPA*** – **B**y**P**ass **A**ctive
- ***CPA*** – **C**hange **P**acket **A**ctive
- ***SPA*** – **S**can **P**acket **A**ctive

60.7.2.1 BPA

The *BPA* state allocates the use of the TMSC pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

60.7.2.2 CPA

The *CPA* state allocates the use of the TMSC pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

60.7.2.3 SPA

The SPA state allocates the use of the TMSC pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

60.7.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMSC pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

60.7.4 Signal behaviors

The TCKC signal:

- The test clock

The TMSC signal:

- An input when the Standard Protocol is used
- Bidirectional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
  - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
  - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

60.7.5 APU Functions

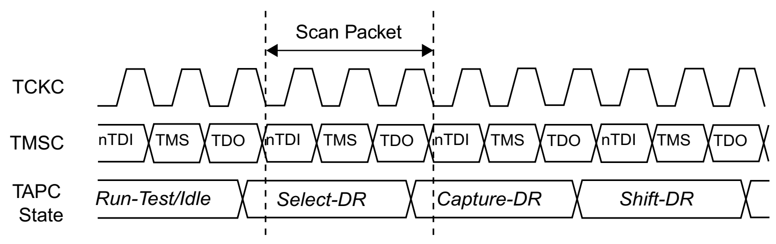
60.7.5.1 Bypass Active (BPA) Function

When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (for example, a narrow TAP.7, or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

60.7.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in *Figure 1236*. There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

Figure 1236. Scan Packet sequence

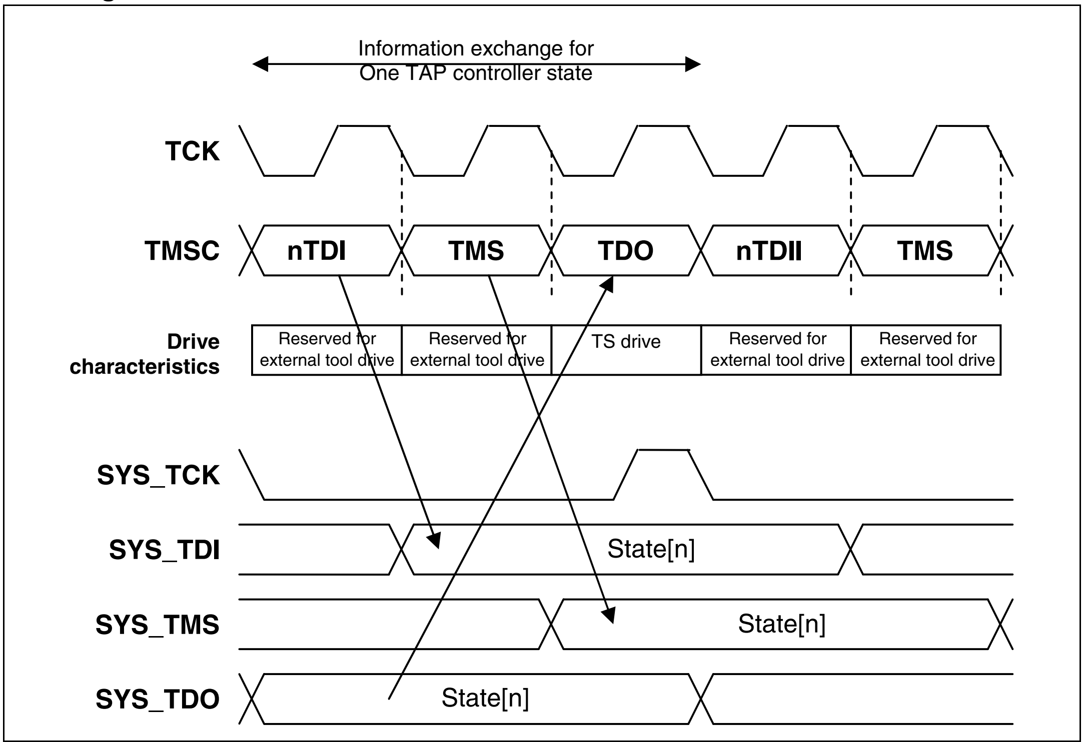


A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

Figure 1237. Serialization and De-serialization of Scan Packet Information



In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats).

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

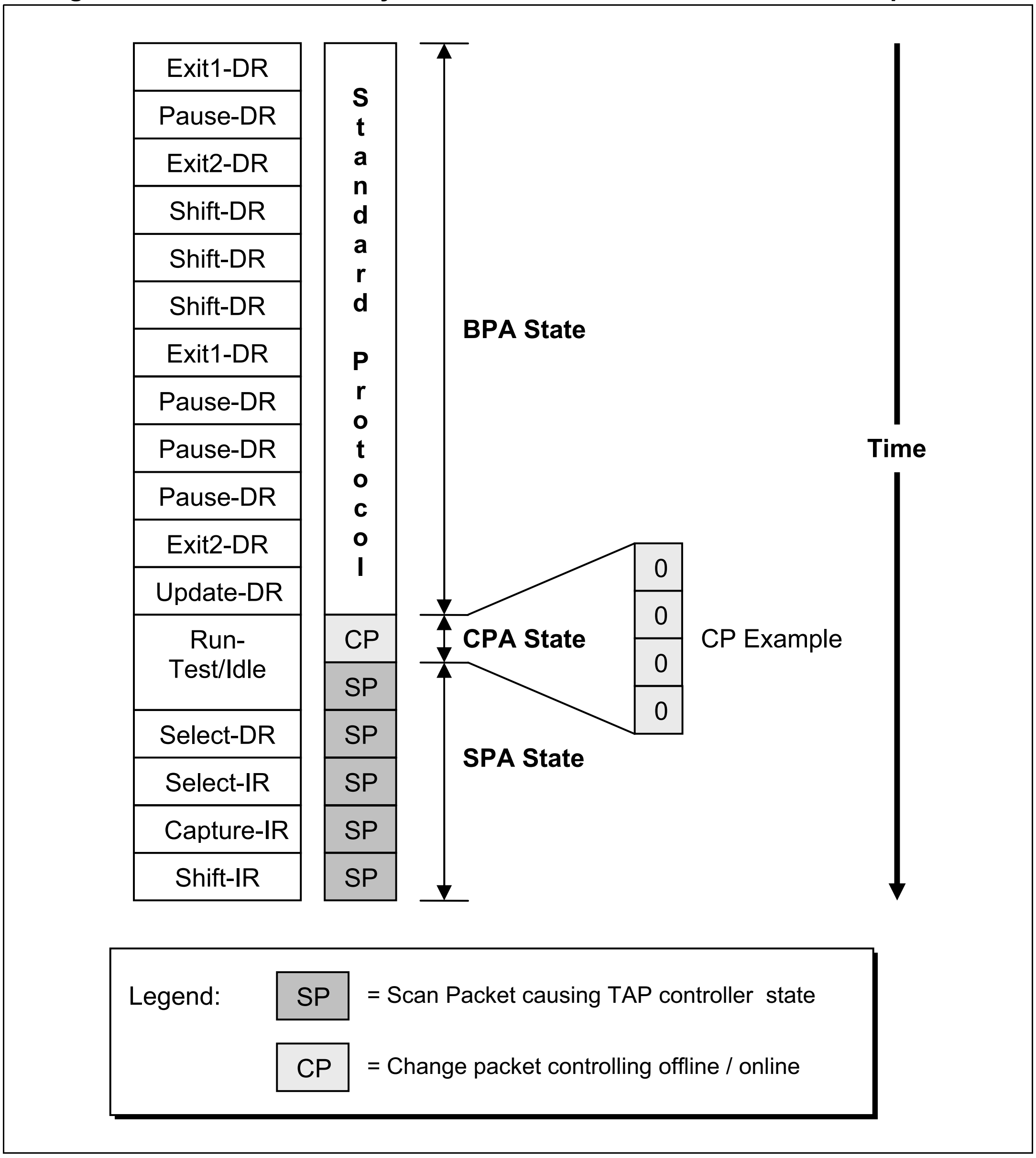
With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

60.7.5.3 Control Process Active (CPA) Function

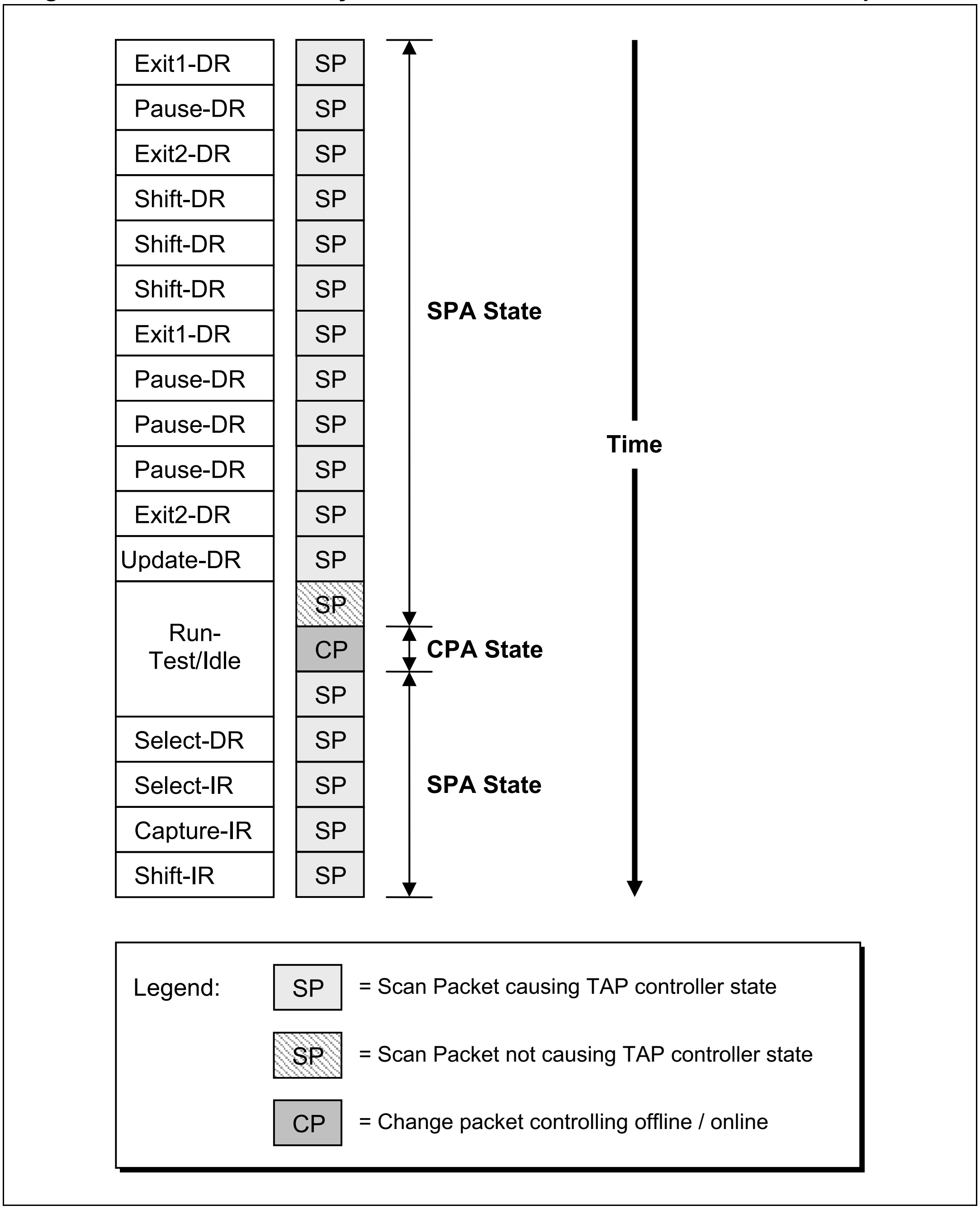
While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

Figure 1238. CPA State Entry from BPA State with Continued Online Operation



An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

Figure 1239. CPA State Entry from the SPA State with Continued Online Operation



When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

Figure 1240. CPA State Entry from the BPA State with Offline Operation

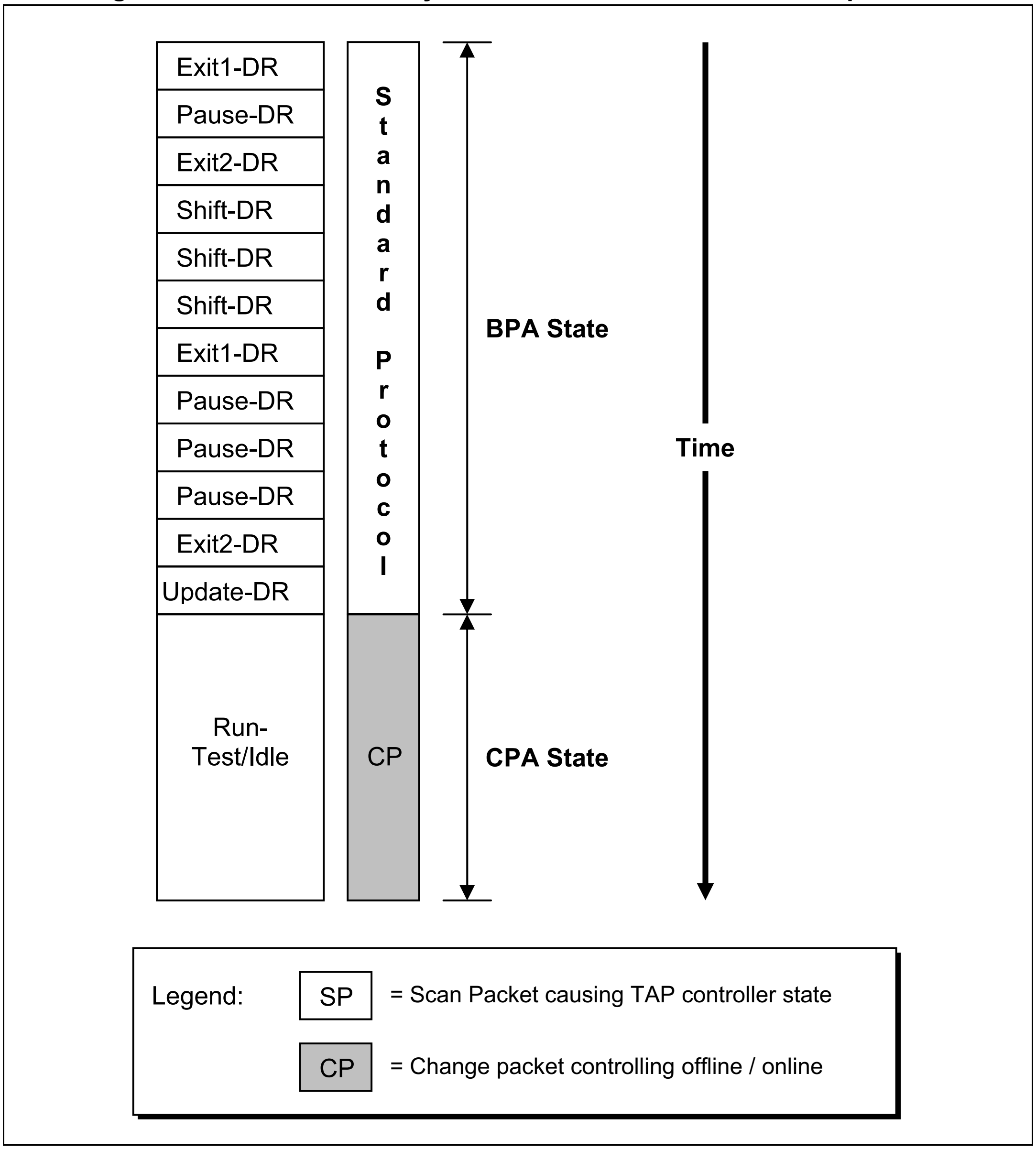
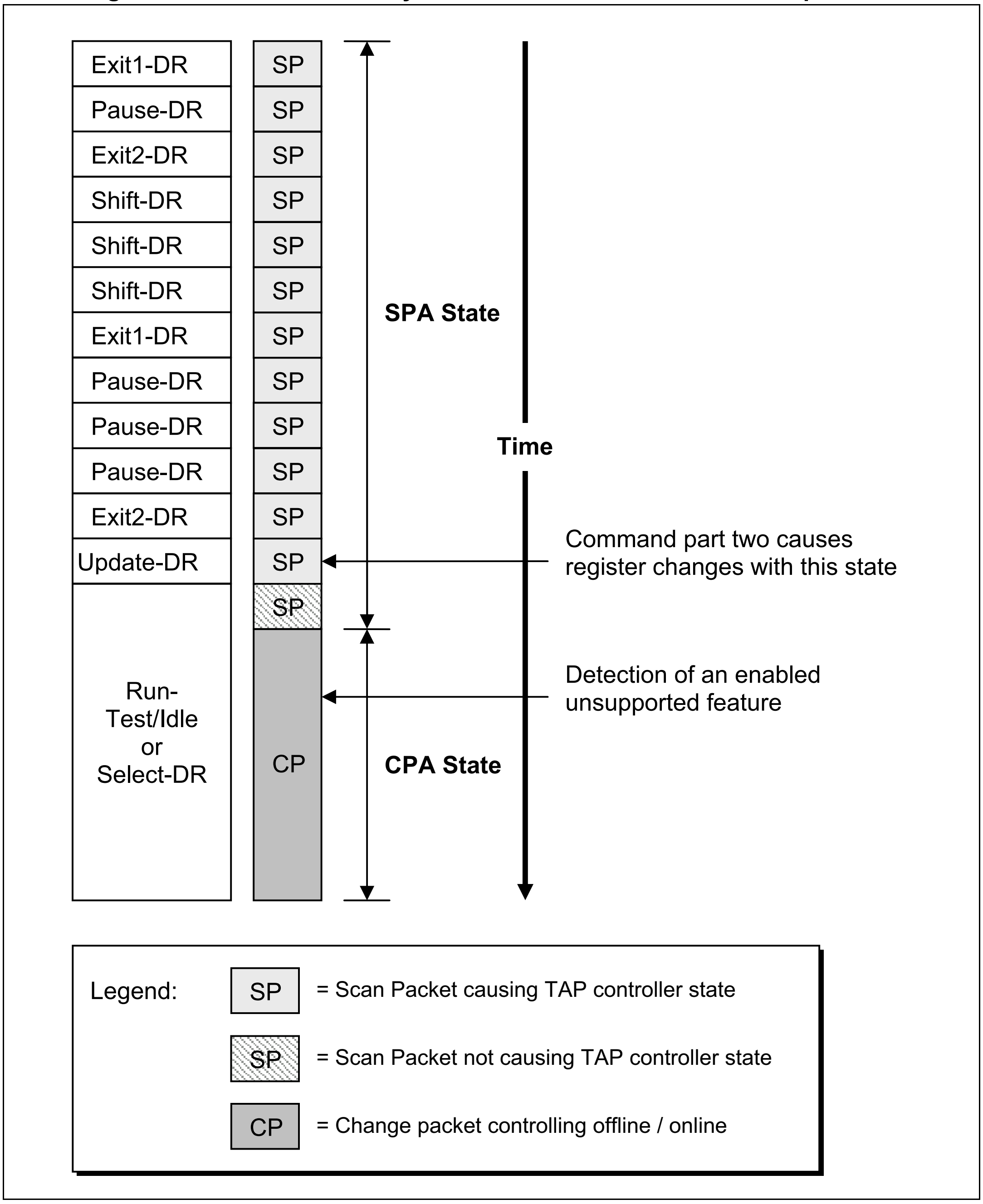


Figure 1241. CPA State Entry from the SPA State with Offline Operation



A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

60.7.6 Configuration Change Packets (CP)

A Change Packet (CP) is composed of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.
- **Body** – Two or more bits representing one or more CP directives:
  - **00** – CP_END
  - **01** – CP_NOP (extends the body by 1-bit)
  - **10** – CP_NOP (extends the body by 1-bit)
  - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the post amble bit)
- **Post amble** – A 1-bit value that is a don't care. This bit facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Post amble. This can be as simple as four logic zero values.

60.7.7 Scan Packet

A Scan Packet defines the period of TMSC pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof).
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

60.7.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

60.7.8.1 Payload Element Content

60.7.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain an input bit frame, an output bit frame, or both. When an input bit frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

60.7.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in *Table 1255*.

Table 1255. Factors determining the SP payload content

Scan format	Payload content determined by the:	
	TAPC state	Header
MScan	–	–
OScan	Yes	–

60.7.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

60.7.8.3 Scan Formats

60.7.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in *Table 1256* and *Table 1258*, respectively. The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

60.7.8.3.2 MScan scan formats

The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

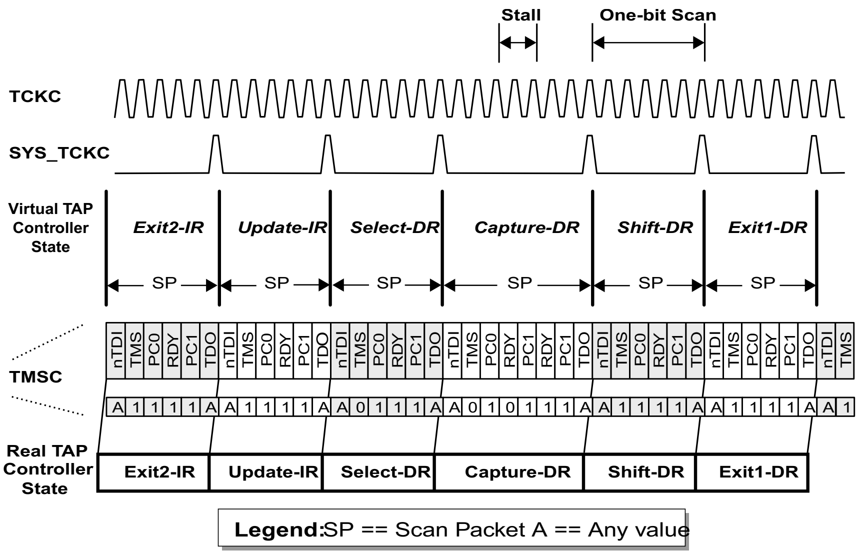
Table 1256. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			
	Multi-chip debug communication			
	Controller ID Allocation			

Table 1257. MScan Scan Packet content

	Non-shift TAP controller states							Shift TAP controller states					
MScan	nTDI	TMS	PC0	RDY	PC1	TDO		nTDI	TMS	PC0	RDY	PC1	TDO

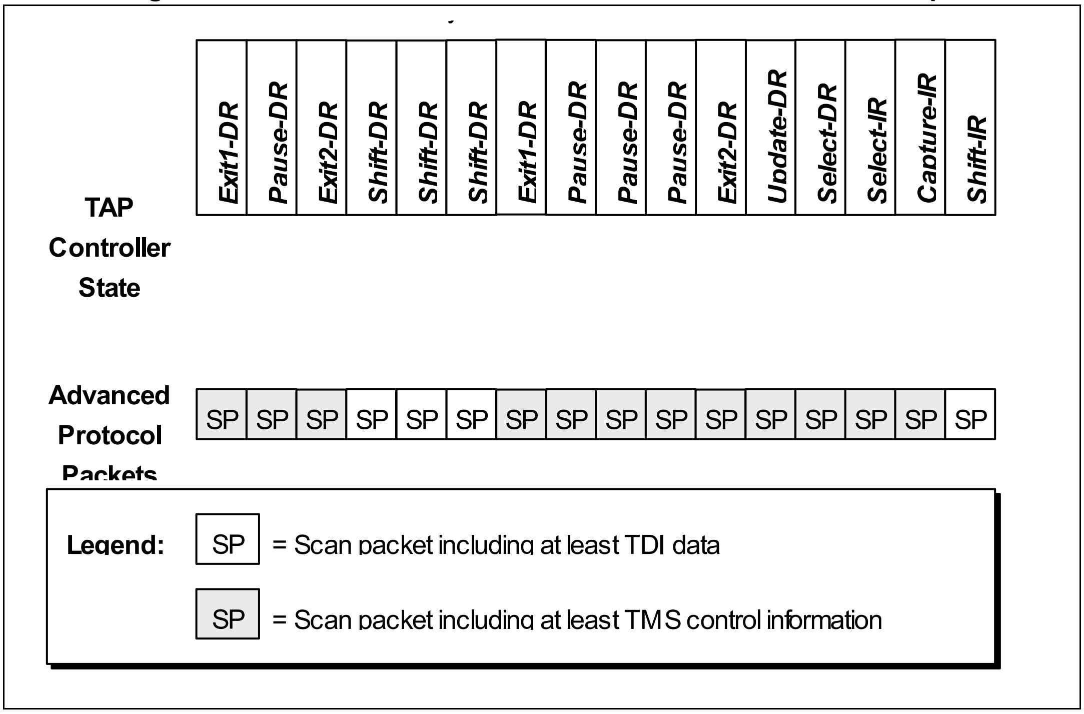
Figure 1242. MScan Transaction



60.7.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

Figure 1243. OScan Scan Packet content/TAPC state relationships



OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan 4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, and EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods for this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in *Table 1258*.

Table 1258. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best		
OScan4	Stalls	↓	Best	External tool or TS
OScan5	Test		↑	
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 1259. OScan Scan Format “Payload”

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	—	TDO		nTDI	TMS	—	TDO
OScan2	—	TMS	—	—		nTDI	TMS	—	TDO
OScan3	—	TMS	—	—		nTDI	TMS	—	—
OScan4	nTDI	TMS	RDY	TDO		nTDI	—	RDY	TDO
OScan5	nTDI	TMS	–	TDO		nTDI	—	—	TDO
OScan6	—	TMS	—	—	Note 1	nTDI	TMS	—	TDO
					Note 2	nTDI	—	—	TDO
OScan7	—	TMS	—	—		nTDI	—	—	—

Note 1: Following Capture-xR or Exit2-xR Scan Packet.
Note 2: Following Shift-xR Scan Packet.

60.8 Functional Description

This section combines the information given in the Ancillary Services, EPU Operation, and APU Operation sections into a single reference for performing common TAP.7 tasks.

60.8.1 Switching from Standard Protocol to Advanced Protocol

Upon exit of Type-0 to Type-4 reset, the CJTAG begins operation in Standard Protocol 4-pin mode. The following steps are required to move from Standard Protocol 4-pin mode to Advanced Protocol (either 4-pin or 2-pin) mode.

The first step required to move to the Advanced Protocol is to change the CJTAG control level to control level 2 via zero-bit scans (ZBS), as described in *Section 60.6.1.4: Control levels*.

After the control level is set to control level 2, TAP.7 commands can be executed as described in *Section 60.6.3: EPU Commands*. Execution of the Store Scan Format (STFMT) to set the scan format to any OSCAN0-7 or MSCAN format begins the switch from the Standard Protocol to the Advanced Protocol.

Once the STFMT command is executed selecting an Advanced Protocol scan format, the CJTAG transitions to the intermediate Control Protocol mode. The Control Protocol uses TMS inputs to control Configuration Change Packets as described in *Section 60.7.5.3: Control Process Active (CPA) Function* and *Section 60.7.6: Configuration Change Packets (CP)*. Loading a change packet body value of 00 (CP_END) results in the switch to the Advanced Protocol.

If Advanced Protocol 2-pin operation is desired, the APFC register should be written to a value of 10 or 11 prior to the execution of the STFMT command that selects an advanced scan format. The APFC register is written via the Store Conditional 2-bit (STC2) command described in *Section 60.6.3.5.1.3: Store Conditional 2-bit (STC2)*.

61 JTAG Data Communication (JDC)

61.1 Introduction

The JTAG Data Communication (JDC) module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory mapped register space and an external tool that accesses the JTAG port.

61.2 Overview

The JDC module consists of IPS accessible registers, JTAG accessible registers, and associated logic to coordinate movement of data from one register domain to another.

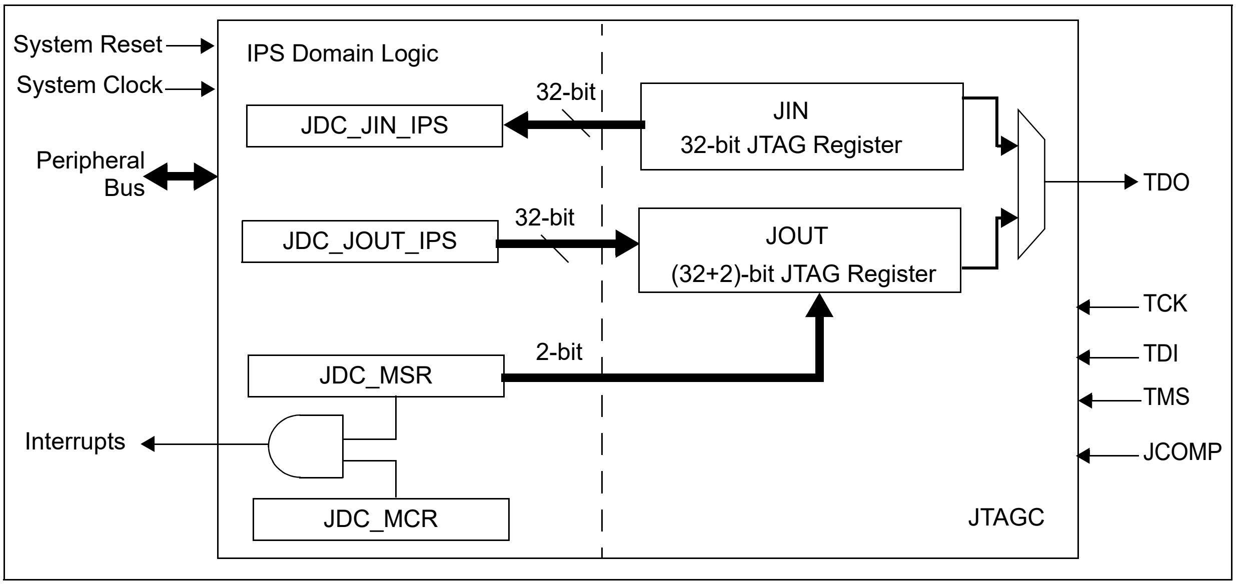
The JDC implements the following IPS data registers, occupying separate memory space:

- 32-bit memory mapped register that can be read or written via IPS (JDC_JOUT_IPS), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 32-bit memory mapped register that can only be read via IPS (JDC_JIN_IPS), and whose contents are loaded from a JTAG register (JIN).

In addition to the data registers themselves, logic is implemented to indicate when new data has been shifted in via the JTAG port and is ready to be read from the JDC_JIN_IPS register, and when new data has been written to the JDC_JOUT_IPS register and is ready to be read via the JTAG port. The state of these flags is captured into a status register (JDC_MSR), and also provided to a JTAG register (JOUT) for tool visibility.

There is a single bus interface to port register data out to the JTAGC, and a single bus interface to port data in from the JTAGC. The architecture is shown in *Figure 1244*.

Figure 1244. JDC block diagram



61.3 Register definition

Table 1260 shows the JDC module registers. Four 32-bit registers are implemented. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

Table 1260. JDC memory map

Address offset	Register	Access	Reset value	Location
0x0	Module Configuration Register (JDC_MCR)	R/W	0x0000_0000	*Section 61.3.1.1*
0x4	Module Status Register (JDC_MSR)	W1C	0x0000_0000	*Section 61.3.1.2*
0x8	JTAG Output IPS Data Register (JDC_JOUT_IPS)	R/W	0x0000_0000	*Section 61.3.1.3*
0xC	JTAG Input IPS Data Register (JDC_JIN_IPS)	Read Only	0x0000_0000	*Section 61.3.1.4*

61.3.1 Register descriptions

61.3.1.1 Module configuration register (JDC_MCR)

Figure 1245 shows the JDC_MCR. The JDC_MCR is used to enable the interrupt outputs of the JDC. This register is reset by system destructive reset.

Offset: 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	JIN_IEN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	JOUT_IEN
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1245. Module Configuration Register (JDC_MCR)

The JDC_MCR is described in *Table 1261*.

Table 1261. JDC_MCR register field descriptions

Field	Description
15 JIN_IEN	JIN Interrupt Enable. 1 Setting of JDC_MSR[JIN_INT] bit asserts the JIN interrupt 0 Setting of JDC_MSR[JIN_INT] bit does not assert the JIN interrupt
31 JOUT_IEN	JOUT Interrupt Enable. 1 Setting of JDC_MSR[JOUT_INT] bit asserts the JOUT interrupt 0 Setting of JDC_MSR[JOUT_INT] bit does not assert the JOUT interrupt

61.3.1.2 Module status register (JDC_MSR)

Figure 1246 shows the format of the JDC_MSR. The JDC_MSR holds the JTAG register status and interrupt bits. This register is reset by system destructive reset.

Offset: 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	JIN_RDY	0	JIN_INT	0	0	0	0	0	0	0	0	0	0	0	0	0	JOUT_RDY	0	JOUT_INT
W																w1c																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1246. Module Status Register (JDC_MSR)

The JDC_MSR is described in *Table 1262*.

Table 1262. JDC_MSR register field descriptions

Field	Description
13 JIN_RDY	JIN Ready (read only). 1 Set when new data is written to the JDC_JIN_IPS register 0 Cleared upon software read of JDC_JIN_IPS contents via IPS
15 JIN_INT	JIN Interrupt. 1 Set when new data is written to the JDC_JIN_IPS register 0 Cleared by writing logic 1
29 JOUT_RDY	JOUT Ready (read only). 1 Set when new data is written to the JDC_JOUT_IPS register 0 Cleared upon tool read of JOUT register via JTAG port
31 JOUT_INT	JOUT Interrupt. 1 Set when JOUT_RDY bit is cleared by tool reading JOUT register 0 Cleared by writing logic 1

61.3.1.3 JTAG output IPS data register (JDC_JOUT_IPS)

The JDC_JOUT_IPS register holds data written via IPS. The JDC_JOUT_IPS contents are ported out and captured into the JOUT register to be read via the JTAG port. *Figure 1247* shows the format of the JDC_JOUT_IPS register. This register is reset by system destructive reset.

Offset: 0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1247. JDC_JOUT_IPS register

The JDC_JOUT_IPS register is described in *Table 1263*.

Table 1263. JDC_JOUT_IPS register field descriptions

Field	Description
0:31 DATA	Contains JOUT_IPS data.

61.3.1.4 JTAG input IPS data register (JDC_JIN_IPS)

Data written to the JTAG input data register (JIN) via the JTAG port is held in the JDC_JIN_IPS register, where it can be read via IPS. *Figure 1248* shows the format of the JDC_JIN_IPS register. This register is reset by system destructive reset.

Offset: 0x0C Access: Read-only

	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
R	DATA
W	
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 1248. JDC_JIN_IPS register

The JDC_JIN_IPS register is described in *Table 1264*.

Table 1264. JDC_JIN_IPS register field descriptions

Field	Description
0:31 DATA	Contains JIN_IPS data.

61.3.2 Non-Memory mapped register definition

The JDC also implements two JTAG accessible registers that are not memory mapped. One JTAG register is used to shift in data to be placed in the JDC_JIN_IPS register. The other JTAG register captures data from the JDC_JOUT_IPS register and ready status from the JDC_MSR to be shifted out via the JTAG port.

61.3.2.1 JTAG output data register (JOUT)

The JOUT register captures data from the JDC_JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JDC_JIN_RDY and JDC_JOUT_RDY status bits. *Figure 1249* shows the format of the JOUT register. This register is reset by system destructive reset and JTAG reset.

Figure 1249. JOUT JTAG register

	33	…	2	1	0
R W	JOUT_IPS Data			JIN_RDY	JOUT_RDY
Reset	*(1)	*(1)	*(1)	*(1)	*(1)

1. The reset value of the JOUT register is 34'b0.

The JOUT register is described in *Table 1265*.

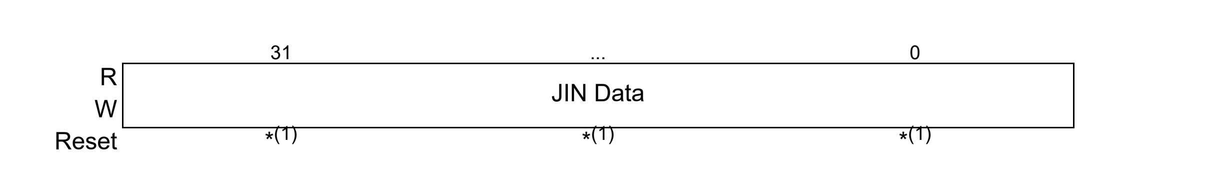
Table 1265. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from JDC_MSR
JOUT_RDY	State of JOUT_RDY bit from JDC_MSR

61.3.2.2 JTAG input data register (JIN)

Data is written to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JIN data is later captured in the JDC_JIN_IPS register to be read via IPS. *Figure 1250* shows the format of the JIN register. This register is reset by system destructive reset and JTAG reset.

Figure 1250. JIN JTAG Register



1. The reset value of the JIN register is 32'b0.

The JIN register is described in *Table 1266*.

Table 1266. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JDC_JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

61.4 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data into a memory mapped register that can be accessed via IPS. It also provides the ability to capture data written to a memory mapped register into a JTAG shift register for output via the JTAG port. An overview of the module functionality is described below.

- A software write to the JOUT_IPS register sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port. The state of the JOUT_RDY bit is reflected in the MSR and also ported out to the JOUT register. A JTAG read of the JOUT register via execution of the JOUT_READ instruction with a JOUT_RDY bit whose value is logic 1 indicates the register contains new data. The JOUT_RDY flag bit is cleared upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.
- A JTAG write to the JIN register via execution of the JIN_READ JTAG instruction updates the contents of the JDC_JIN_IPS register upon exit of the Update-DR state. An update of the JDC_JIN_IPS register sets the JIN_RDY flag bit, indicating new data

is available to be read via IPS. The state of the JIN_RDY bit is reflected in the MSR register and also ported out to the JOUT register. The JIN_RDY flag bit is cleared upon software read of the JDC_JIN_IPS register. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates new data can be written to the JIN register.

61.4.1 IPS register access

IPS access is available to the registers described in *Section 61.3.1*.

61.4.2 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

61.4.2.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in *Table 1267*. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 1267. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JIN_IPS upon exit of Update-DR state while JIN_WRITE is active.
BYPASS	11111	Selects bypass register for data operations
Reserved	All other opcodes	Decoded to select bypass register

61.5 Use case example

A summary of how the JDC can be used to interact with the Hardware Security Module (HSM) to perform challenge/response password authorization is as follows:

1. After exit of reset, all registers are at their reset values. JIN_RDY bit value of 0 indicates the tool may write data to the JIN register. JOUT_RDY bit value of 0 indicates that software may write data to the JOUT_IPS register.
2. The JDC_MCR register is programmed to enable JOUT and JIN interrupts, or not.
3. Once the tool is ready to start the authorization process, it writes a value to the JIN register via execution of the JTAGC JIN_WRITE instruction.
4. Software monitors the state of the JIN_RDY bit or enables the JIN interrupt and uses the interrupt assertion as a trigger to start the authorization process.
5. Once the authorization process is started, software writes the first JOUT value to the JOUT_IPS register. This sets the JOUT_RDY bit value to 1.
6. Following the initial tool write to the JIN register, the tool polls the JOUT register to determine when software has provided a challenge word. Upon reading the JOUT register with JOUT_RDY bit set, the tool records the data value to be used to generate

the appropriate response. The read of the JOUT register clears the JOUT_RDY bit. The clearing of the JOUT_RDY bit also sets the JOUT_INT bit and asserts the JOUT interrupt if the JDC_MCR[JOUT_IEN] is set.

7. With the JOUT_RDY bit cleared, software may now provide the next challenge word.
8. Once the tool has generated a response word, it uses the value of the JIN_RDY bit from the JOUT register read to determine when it can perform a write to the JIN register to provide the response.
9. The HSM can provide additional challenge words and read back the corresponding response words as needed, repeating the process.

Simply providing a password without using the challenge/response protocol can be done in a similar way. The tool provides 32-bits of the password at a time with each write to the JIN register, and monitors the JIN_RDY bit value of the JOUT register to determine when the next JIN register value can be written. There is no need for software to write the JOUT_IPS register in this case.

62 Sequence Processing Unit (SPU)

62.1 Introduction

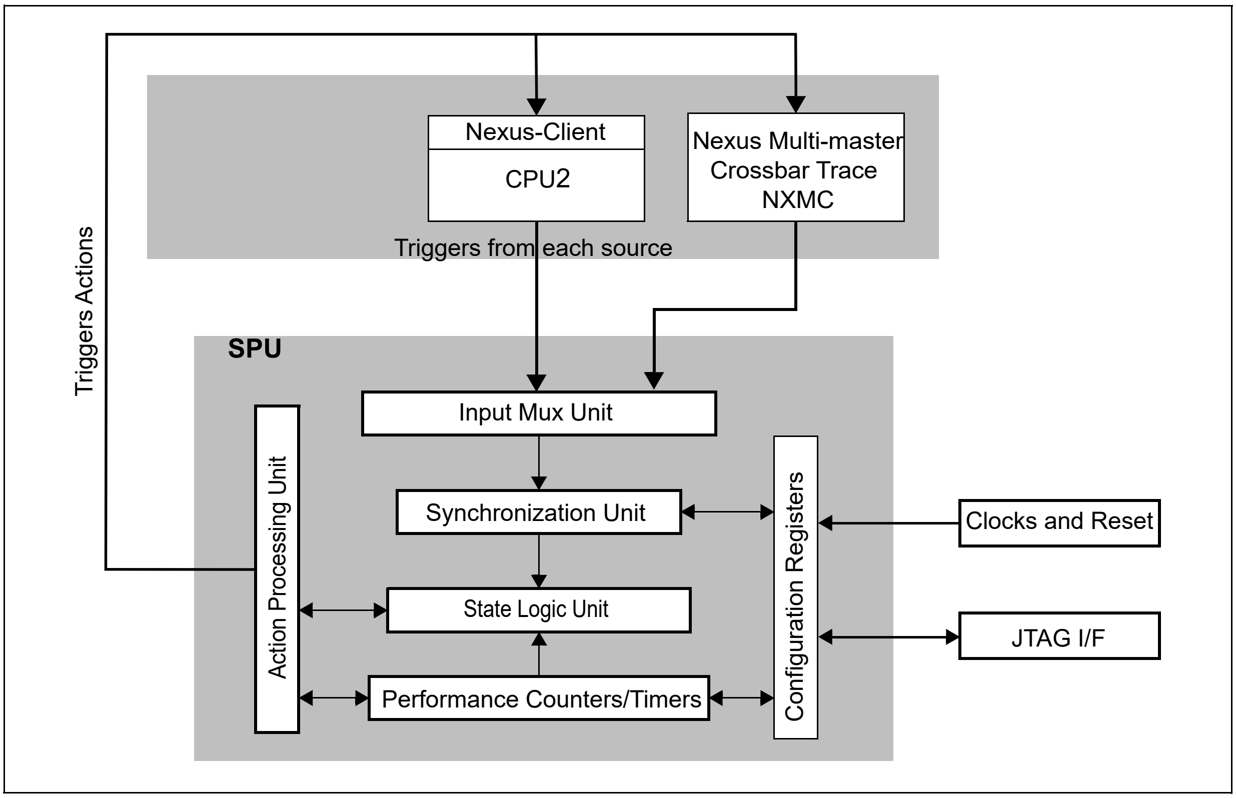
The Sequence Processing Unit (SPU) provides an on-device trigger functions similar to those found on a logic analyzer. In a complex SoC, performance monitor functions are available at two levels: system level and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System level performance monitor functions are integrated into the SPU complex trigger logic, and thus allow counting and timing of any debug trigger combinations supported by the SPU.

Various clients such as the cores or crossbar switch, generate watchpoints and triggers when operating in their debug mode. The SPU collects these triggers (such as interrupt occurrence, address watchpoint, and so on) and uses them as conditions to sequence through states, with resultant actions (such as start/stop trace, start/stop counter, capture timebase, and so on). The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, complex logic-analyzer-like behavior can be achieved, providing vital real-time visibility and debug ability of the activities of the system.

62.1.1 SPU block diagram

Figure 1251 shows a top-level block diagram of the SPU.

Figure 1251. SPU block diagram



62.1.2 Overview

The SPU is able to generate complex debug events, based upon input events from sources (CPUs, and system bus clients) throughout the SoC. The SPU has the ability to create a state machine to trigger a debug action. Single or multiple actions can be triggered by a state machine, which results in debug events being created. There are a selection of timers and counters available to assist comparison events.

Complex trigger support is implemented in the State Logic Unit (SLU). There are numerous inputs coming from different clients. Users can choose 63 inputs from a set of 209 inputs at the level one Mux. There are actually 64 inputs to the Muxes, but one of the inputs is permanently connected to a high level so users can only choose 63 of the inputs. Of these 63 inputs, users can select 16 inputs for each SLU, shown in *Figure 1286*.

One SLU consists of four input AND gates, the output of which are ORed together. One of the inputs of each AND gate has the option to invert it. Also, the output of each AND gate can be optionally inverted. Finally, the output of each SLU has the option of inversion. The block diagram for one SLU state is shown in *Figure 1287*. The SPU includes eight SLUs states.

Users can utilize these states to form a configurable state machine. This allows users to join states together with the if-then-else operations to create a sequence. Each state in a sequence has the ability to move to another state based on the true condition from the state logic, and the ability to route to another state based on the a false condition from the state logic. This is shown in *Figure 1288* and *Figure 1289*. At a given time, a maximum of four sequences are allowed. Each sequence can have one state or maximum of eight states. Each state can only be used in one unique sequence. The number of states in all the sequences cannot exceed eight. Each state has the ability to trigger one to four actions based on a true/false condition in the active sequence.

An example of typical complex trigger for the SPU would be: if FunctionA is entered, monitor CPU writes to VariableB. If CPU writes a value to VariableB which is 0x100, then enable trace for the halt CPU.

The SPU consists of the following sub-blocks:

- Input Mux Unit: receives the numerous triggers from various clients and reduces them down into a user selected set. The muxing unit selects sixteen inputs for each sequencing SLU from a set of 63 static inputs. This selection is done using two levels of muxing. At the first level of muxing, users select 63 out of 209 inputs. At the second level of muxing, out of these 63 inputs, one input can be selected for each of the four input AND gates. There are 128 64 $\times$ 1 Muxes to cater the needs of 128 inputs of all the eight states. This input mux selection logic is shown in *Figure 1286*.
- State Logic Units (SLU): responsible for performing operations on the selected input events to produce desired events. Each SLU has a control register that manages how the various inputs are combined to produce a desired event. Only one input of each four input AND gate can be independently inverted. The combined event can also be optionally inverted and, coupled with DeMorgan’s law, allows for generalized signal combinations. This is shown in *Figure 1287*.
- Action Processing Unit: takes an action request from a SLU state and converts this into one or multiple actions. The action unit can be programmed to trigger various actions in response to each processed event from the SLU unit. The user can define the actions associated with each state from each SLU.
- Counters/Timers: the SPU implements 16 32-bit timer/counter functionality. Timer or counter selection is done via a configurable register in the SPU. Each counter/timer

has its own control register. A 32-bit comparator value can optionally be compared against the timer/counter. These registers are accessible via JTAG.

62.1.3 Features

The features of the SPU are as follows:

- clients:
  - CPU
  - 1 System Bus clients (NXMC)
- Input watchpoints/triggers selection
- Input debug triggers for instruction execution selection
- Input interrupt selection
- Sequences and states criteria
  - Up to 4 user programmable number of active/simultaneous sequences
  - Up to 8 user programmable states that can be split between all four sequences
  - Each state can only be used in one unique sequence
  - A sequence can consist of a single state or any number of states up to 8
- State logic
  - Each state logic consist of sum of products (SOP)
  - SOP consist of four 4-input AND gates, and one 4-input OR gate
  - Optional inversion on one input of each AND gate of each state logic
  - Optional inversion on output of each AND gate
  - Optional inversion on output of OR gate of each state
  - States are joined together with if-then-else statement to create a sequence
- SLU events
  - Up to 8 user programmable derived events/actions
  - Any state can optionally trigger of 1 to 4 actions based on a true condition
  - Any state can optionally trigger of 1 to 4 actions based on a false condition
  - User selection of trigger actions generated in response to each programmed trigger
- SPU counters/timers
  - 16 32-bit counters/timers
  - Timer/counter configurability selection via a register
  - Start/increment when used as timer/counter respectively
  - Reset when used as timer/counter respectively
  - Counter value compare match
  - Counter overflow event status
  - Generation of a greater than condition against a 32-bit comparator value (no support for range condition from single counter)
  - Range support using two counters
  - Use of counter/timer registers to generate counted or timed triggers
  - Use of counter/timer registers to capture system performance statistics
  - Each timer/counter value has the ability to be inserted into the Nexus trace stream upon an SPU action

- Access to timer/counter values via JTAG
- Prescaler for timer mode only
  - Configurable prescaler of /1, /4, /16 or /32 of the clock for timer operation inside the SPU
- Asynchronous interfaces
  - SPU to/from CPU (fastest clock)
  - SPU to/from NXMC
  - SPU to/from DCI
  - SPU to/from JTAGM
- Interfaces
  - CPU
  - System bus
  - DCI
- Configuration registers: 32-bit double word aligned configuration registers (byte and halfword access not supported)
- SPU actions: SPU block triggers various actions based on the derived events, these actions are defined in *Section 62.2*

62.2 SPU actions

Sequence actions describe what is triggered. A sequence state may only have a GoTo action to determine the next sequence state. The complete set of actions are as follows:

- Start/stop trace for a source
- Start/stop/increment counter/timer counter/timer[0–15]
- Reset counter/timer[0–15]
- Halt1: single signal (EVTI0) that can be used to halt a pre-configured DCI group1
- Single interrupt for INTC to interrupt a CPU: de-assert (cleared) via JTAGM
- Generate a pulse on the EVTO pin: output pulse over EVTO1 and EVTO2
- Start or Stop Performance Counters (PMC) for a CPU source: Start/Stop CPUn
- Optional trace group 1 (Start/Stop Trace a pre-configured set of trace signals): four (DTM,PTM,WTM,OTM) independent start/stop trace groups can be formed. One or more clients (CPUn and Nexus) can be part of a particular group.

Note: GoTo is not part of the Action list described above but it is implemented in the state logic to move from one state to another. If there is nothing defined for GoTo, it signifies the end of the sequence.

In the condition where a state (State n) updates a counter and the next state in the sequence (State n+1) evaluates the value of the same counter (for example, to compare if the counter is less-than a certain value), the update to the counter (from State n) is not applied before the counter evaluation in the next state (State n+1). This results in the old value of the counter being evaluated (in State n+1), rather than the desired updated value. This is due to hardware limitations of the SPU as the update to the counter does not complete until the clock cycle after the evaluation of the next state. This limitation is only applicable in Div1 mode and not in Div4 mode.

To workaround this hardware limitation, a delay should be inserted between the update to a counter and the evaluation of this counter. A dummy state can be used between the counter update (State n) and the counter evaluation (State n+1). A dummy state is a state with any input, that on both then/else conditions, the action proceeds to the next state (State n+1) where the counter evaluation occurs. The insertion of this dummy state provides sufficient time for the update of the counter to be applied, in time for the counter evaluation. The same outcome is also possible in a complex sequence with multiple states by scheduling states so that counter update and counter evaluation are not performed on consecutive states in a sequence.

62.3 Operation mode

62.3.1 SPU enable/disable

By default, the SPU is in disabled mode. In this state:

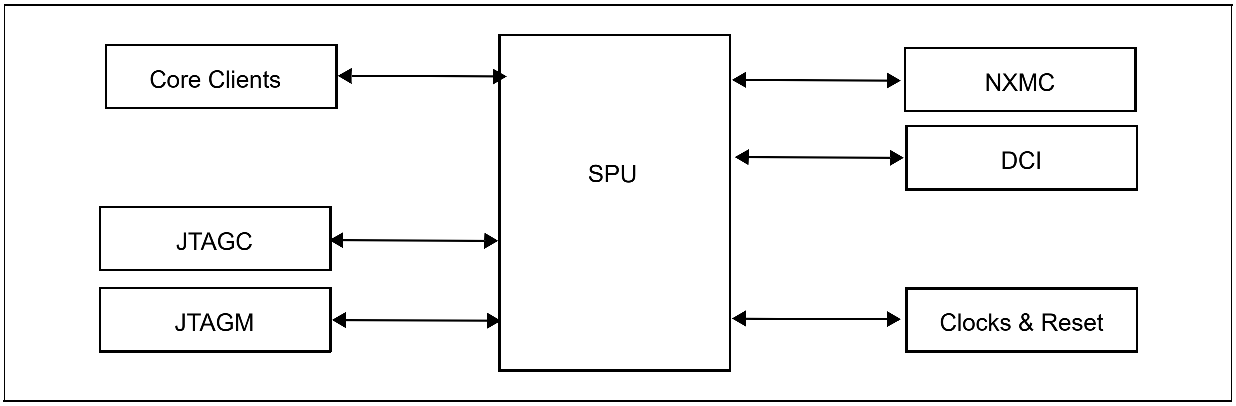
- All events/inputs are disabled such that no counting or other actions occur.
- All the SPU control registers are accessible.

The user must configure the control registers before enabling the SPU via the SPU Enable/Disable register.

62.4 SPU module interface

The SPU is connected to many clients such as the cores, system bus and Nexus clients. The SPU also interfaces to the DCI and the JTAG modules as shown in *Figure 1252*.

Figure 1252. SPU module interface



62.5 Register definition

The SPU contains 32-bit aligned registers (byte and half-word access are not supported). All configuration registers in the SPU are JTAG registers such that these registers are accessible via JTAG interface only. Write access to reserved locations have no impact while read access to reserved locations always return 0. These registers have Nexus reset and JTAG reset control.

Note: The SPU does not check for correctness of the programmed values in the registers and programmers must ensure that correct values are being written.

Table 1268 lists the memory mapped registers which control the operation of the SPU. There are two instructions for the SPU registers:

- Instruction to enable the SLU (first set of registers with offsets 0x01–0x49) registers is 4'h3.
- Instruction to enable the Counter Control Unit (second set of registers with offsets 0x01–0x72) is 4'h2.

Table 1268. SPU register summary

Offset	Register	Location
0x01	Level1 Mux Selection 0 (L1SEL0)	*Section 62.5.1.1.1*
0x02	Level1 Mux Selection 1 (L1SEL1)	*Section 62.5.1.1.2*
0x03	Level1 Mux Selection 2 (L1SEL2)	*Section 62.5.1.1.3*
0x04	Level1 Mux Selection 3 (L1SEL3)	*Section 62.5.1.1.4*
0x05	Level1 Mux Selection 4(L1SEL4)	*Section 62.5.1.1.5*
0x06	Level1 Mux Selection 5 (L1SEL5)	*Section 62.5.1.1.6*
0x07	Level1 Mux Selection 6 (L1SEL6)	*Section 62.5.1.1.7*
0x08	Level1 Mux Selection 7 (L1SEL7)	*Section 62.5.1.1.8*
0x09–0x0A	Reserved for future use	
0x0B	Input Trigger Level Detection 2 (ITLD2)	*Section 62.5.1.1.9*
0x0C	Input Trigger Level Detection 3 (ITLD3)	*Section 62.5.1.1.10*
0x0D	Reserved	
0x0E	Reserved	
0x0F	CPU2 Processor Except Vector Prefix (C2PEVP)	*Section 62.5.1.1.11*
0x10–0x11	Reserved for future use	
12	Reserved	
13	Reserved	
14	CPU2 Interrupt Priority Selection (C2PIS)	*Section 62.5.1.1.12*
0x15–0x16	Reserved for future use	
0x17	Level2 Mux State0 Selection 0 (L20SEL0)	*Section 62.5.1.2.1*
0x18	Level2 Mux State0 Selection 1 (L20SEL1)	*Section 62.5.1.2.2*
0x19	Level2 Mux State0 Selection 2 (L20SEL2)	*Section 62.5.1.2.3*
0x1A	Level2 Mux State0 Selection 3 (L20SEL3)	*Section 62.5.1.2.4*
0x1B	Level2 Mux State1 Selection 0 (L21SEL0)	*Section 62.5.1.2.1*
0x1C	Level2 Mux State1 Selection 1 (L21SEL1)	*Section 62.5.1.2.2*
0x1D	Level2 Mux State1 Selection 2 (L21SEL2)	*Section 62.5.1.2.3*
0x1E	Level2 Mux State1 Selection 3 (L21SEL3)	*Section 62.5.1.2.4*
0x1F	Level2 Mux State2 Selection 0 (L22SEL0)	*Section 62.5.1.2.1*

Table 1268. SPU register summary (continued)

Offset	Register	Location
0x20	Level2 Mux State2 Selection 1 (L22SEL1)	*Section 62.5.1.2.2*
0x21	Level2 Mux State2 Selection 2 (L22SEL2)	*Section 62.5.1.2.3*
0x22	Level2 Mux State2 Selection 3 (L22SEL3)	*Section 62.5.1.2.4*
0x23	Level2 Mux State3 Selection 0 (L23SEL0)	*Section 62.5.1.2.1*
0x24	Level2 Mux State3 Selection 1 (L23SEL1)	*Section 62.5.1.2.2*
0x25	Level2 Mux State3 Selection 2 (L23SEL2)	*Section 62.5.1.2.3*
0x26	Level2 Mux State3 Selection 3 (L23SEL3)	*Section 62.5.1.2.4*
0x27	Level2 Mux State4 Selection 0 (L24SEL0)	*Section 62.5.1.2.1*
0x28	Level2 Mux State4 Selection 1 (L24SEL1)	*Section 62.5.1.2.2*
0x29	Level2 Mux State4 Selection 2 (L24SEL2)	*Section 62.5.1.2.3*
0x2A	Level2 Mux State4 Selection 3 (L24SEL3)	*Section 62.5.1.2.4*
0x2B	Level2 Mux State5 Selection 0 (L25SEL0)	*Section 62.5.1.2.1*
0x2C	Level2 Mux State5 Selection 1 (L25SEL1)	*Section 62.5.1.2.2*
0x2D	Level2 Mux State5 Selection 2 (L25SEL2)	*Section 62.5.1.2.3*
0x2E	Level2 Mux State6 Selection 3 (L25SEL3)	*Section 62.5.1.2.4*
0x2F	Level2 Mux State6 Selection 0 (L26SEL0)	*Section 62.5.1.2.1*
0x30	Level2 Mux State6 Selection 1 (L26SEL1)	*Section 62.5.1.2.2*
0x31	Level2 Mux State6 Selection 2 (L26SEL2)	*Section 62.5.1.2.3*
0x32	Level2 Mux State6 Selection 3 (L26SEL3)	*Section 62.5.1.2.4*
0x33	Level2 Mux State7 Selection 0 (L27SEL0)	*Section 62.5.1.2.1*
0x34	Level2 Mux State7 Selection 1 (L27SEL1)	*Section 62.5.1.2.2*
0x35	Level2 Mux State7 Selection 2 (L27SEL2)	*Section 62.5.1.2.3*
0x36	Level2 Mux State7 Selection 3 (L27SEL3)	*Section 62.5.1.2.4*
0x37	Input/Output Inversion Control State0 (IOIC0)	*Section 62.5.1.3.1*
0x38	Input/Output Inversion Control State1 (IOIC1)	*Section 62.5.1.3.1*
0x39	Input/Output Inversion Control State2 (IOIC2)	*Section 62.5.1.3.1*
0x3A	Input/Output Inversion Control State3 (IOIC3)	*Section 62.5.1.3.1*
0x3B	Input/Output Inversion Control State4 (IOIC4)	*Section 62.5.1.3.1*
0x3C	Input/Output Inversion Control State5 (IOIC5)	*Section 62.5.1.3.1*
0x3D	Input/Output Inversion Control State6 (IOIC6)	*Section 62.5.1.3.1*
0x3E	Input/Output Inversion Control State7 (IOIC7)	*Section 62.5.1.3.1*
0x3F	Sequence Control (SCTRL)	*Section 62.5.1.3.2*
0x40	State0 Goto Control (S0GC)	*Section 62.5.1.3.3*
0x41	State1 Goto Control (S1GC)	*Section 62.5.1.3.3*
0x42	State2 Goto Control (S2GC)	*Section 62.5.1.3.3*

Table 1268. SPU register summary (continued)

Offset	Register	Location
0x43	State3 Goto Control (S3GC)	Section 62.5.1.3.3
0x44	State4 Goto Control (S4GC)	Section 62.5.1.3.3
0x45	State5 Goto Control (S5GC)	Section 62.5.1.3.3
0x46	State6 Goto Control (S6GC)	Section 62.5.1.3.3
0x47	State7 Goto Control (S7GC)	Section 62.5.1.3.3
0x48	SLU Status (SS)	Section 62.5.1.4
0x49	SPU Enable (SE)	Section 62.5.1.5
0x01	State0 True Action 0 (S0TA0)	Section 62.5.1.7.1
0x02	State0 True Action 1 (S0TA1)	Section 62.5.1.7.2
0x03	State1 True Action 0 (S1TA0)	Section 62.5.1.7.1
0x04	State1 True Action 1 (S1TA1)	Section 62.5.1.7.2
0x05	State2 True Action 0 (S2TA0)	Section 62.5.1.7.1
0x06	State2 True Action 1 (S2TA1)	Section 62.5.1.7.2
0x07	State3 True Action 0 (S3TA0)	Section 62.5.1.7.1
0x08	State3 True Action 1 (S3TA1)	Section 62.5.1.7.2
0x09	State4 True Action 0 (S4TA0)	Section 62.5.1.7.1
0x0A	State4 True Action 1 (S4TA1)	Section 62.5.1.7.2
0x0B	State5 True Action 0 (S5TA0)	Section 62.5.1.7.1
0x0C	State5 True Action 1(S5TA1)	Section 62.5.1.7.2
0x0D	State6 True Action 0 (S6TA0)	Section 62.5.1.7.1
0x0E	State6 True Action 1(S6TA1)	Section 62.5.1.7.2
0x0F	State7 True Action 0 (S7TA0)	Section 62.5.1.7.1
0x10	State7 True Action 1(S7TA1)	Section 62.5.1.7.2
0x11	State0 False Action 0 (S0FA0)	Section 62.5.1.7.3
0x12	State0 False Action 1 (S0FA1)	Section 62.5.1.7.4
0x13	State1 False Action 0 (S1FA0)	Section 62.5.1.7.3
0x14	State1 False Action 1(S1FA1)	Section 62.5.1.7.4
0x15	State2 False Action 0 (S2FA0)	Section 62.5.1.7.3
0x16	State2 False Action 1(S2FA1)	Section 62.5.1.7.4
0x17	State3 False Action 0 (S3FA0)	Section 62.5.1.7.3
0x18	State3 False Action 1(S3FA0)	Section 62.5.1.7.4
0x19	State4 False Action 0 (S4FA0)	Section 62.5.1.7.3
0x1A	State4 False Action 1 (S4FA1)	Section 62.5.1.7.4
0x1B	State5 False Action 0 (S5FA0)	Section 62.5.1.7.3
0x1C	State5 False Action 1 (S5FA1)	Section 62.5.1.7.4

Table 1268. SPU register summary (continued)

Offset	Register	Location
0x1D	State6 False Action 0 (S6FA0)	*Section 62.5.1.7.3*
0x1E	State6 False Action 1 (S6FA1)	*Section 62.5.1.7.4*
0x1F	State7 False Action 0 (S7FA0)	*Section 62.5.1.7.3*
0x20	State7 False Action 1 (S7FA1)	*Section 62.5.1.7.4*
0x21	DTM Trace Group Configuration (DTMTGC)	*Section 62.5.1.7.5*
0x22	PTM Trace Group Configuration (PTMTGC)	*Section 62.5.1.7.5*
0x23	OTM Trace Group Configuration (OTMTGC)	*Section 62.5.1.7.5*
0x24	WTM Trace Group Configuration (WTMTGC)	*Section 62.5.1.7.5*
0x25	Interrupt Status (INTS)	*Section 62.5.1.7.6*
0x26	Counter Control 0 (CCTRL0)	*Section 62.5.1.8*
0x27	Counter Control 1 (CCTRL1)	*Section 62.5.1.8*
0x28	Counter Control 2 (CCTRL2)	*Section 62.5.1.8*
0x29	Counter Control 3 (CCTRL3)	*Section 62.5.1.8*
0x2A	Counter Control 4 (CCTRL4)	*Section 62.5.1.8*
0x2B	Counter Control 5 (CCTRL5)	*Section 62.5.1.8*
0x2C	Counter Control 6 (CCTRL6)	*Section 62.5.1.8*
0x2D	Counter Control 7 (CCTRL7)	*Section 62.5.1.8*
0x2E	Counter Control 8 (CCTRL8)	*Section 62.5.1.8*
0x2F	Counter Control 9 (CCTRL9)	*Section 62.5.1.8*
0x30	Counter Control 10 (CCTRL10)	*Section 62.5.1.8*
0x31	Counter Control 11 (CCTRL11)	*Section 62.5.1.8*
0x32	Counter Control 12 (CCTRL12)	*Section 62.5.1.8*
0x33	Counter Control 13 (CCTRL13)	*Section 62.5.1.8*
0x34	Counter Control 14 (CCTRL14)	*Section 62.5.1.8*
0x35	Counter Control 15 (CCTRL15)	*Section 62.5.1.8*
0x36	Counter Compare 0(CCMP0)	*Section 62.5.1.9*
0x37	Counter Compare 1 (CCMP1)	*Section 62.5.1.9*
0x38	Counter Compare 2 (CCMP2)	*Section 62.5.1.9*
0x39	Counter Compare 3 (CCMP3)	*Section 62.5.1.9*
0x3A	Counter Compare 4 (CCMP4)	*Section 62.5.1.9*
0x3B	Counter Compare 5 (CCMP5)	*Section 62.5.1.9*
0x3C	Counter Compare 6 (CCMP6)	*Section 62.5.1.9*
0x3D	Counter Compare 7 (CCMP7)	*Section 62.5.1.9*
0x3E	Counter Compare 8 (CCMP8)	*Section 62.5.1.9*
0x3F	Counter Compare 9 (CCMP9)	*Section 62.5.1.9*

Table 1268. SPU register summary (continued)

Offset	Register	Location
0x40	Counter Compare 10 (CCMP10)	*Section 62.5.1.9*
0x41	Counter Compare 11 (CCMP11)	*Section 62.5.1.9*
0x42	Counter Compare 12 (CCMP12)	*Section 62.5.1.9*
0x43	Counter Compare 13 (CCMP13)	*Section 62.5.1.9*
0x44	Counter Compare 14 (CCMP14)	*Section 62.5.1.9*
0x45	Counter Compare 15 (CCMP15)	*Section 62.5.1.9*
0x46	Counter Compare Status (CCOMS)	*Section 62.5.1.10.1*
0x47	Counter Overflow Status (COS)	*Section 62.5.1.10.2*
0x48	Counter Capture Status (CCAPS)	*Section 62.5.1.10.3*

62.5.1 Registers description

62.5.1.1 Level1 input Mux configurations registers

A set of input Mux control registers is used to narrow down the large number of SPU watchpoints to a manageable set. There are sixty-four 8 × 1 (eight inputs and one output) multiplexers. Three bits per Mux are used to select one input out of 8 inputs. 8 Muxes are configured with each 32-bit register; in other words, 8 inputs out of 64 for each register. This provides the flexibility to the programmer to route any input trigger to any state input. The detail of this selection is shown in *Figure 1286*.

Note: Users cannot program MUX 63 in L1SEL7 (the 64th mux). All eight inputs of the 63rd mux are tied to high and the output is always high (1'b1).

62.5.1.1.1 Level1 Mux Selection 0 (L1SEL0)

Figure 1253 shows the format of the L1SEL0 register. The values for the fields of the L1SEL0 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x01 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 7			0	MUX 6			0	MUX 5			0	MUX 4		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 3			0	MUX 2			0	MUX 1			0	MUX 0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1253. Level1 Mux Selection 0 (L1SEL0)

62.5.1.1.2 Level1 Mux Selection 1 (L1SEL1)

Figure 1254 shows the format of the L1SEL1 register. The values for the fields of the L1SEL1 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x02 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 15			0	MUX 14			0	MUX 13			0	MUX 12		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 11			0	MUX 10			0	MUX 9			0	MUX 8		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1254. Level1 Mux Selection 1 (L1SEL1)

62.5.1.1.3 Level1 Mux Selection 2 (L1SEL2)

Figure 1255 shows the format of the L1SEL2 register. The values for the fields of the L1SEL2 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x03 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 23			0	MUX 22			0	MUX 21			0	MUX 20		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 19			0	MUX 18			0	MUX 17			0	MUX 16		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1255. Level1 Mux Selection 2 (L1SEL2)

62.5.1.1.4 Level1 Mux Selection 3 (L1SEL3)

Figure 1256 shows the format of the L1SEL3 register. The values for the fields of the L1SEL3 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x04 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 31			0	MUX 30			0	MUX 29			0	MUX 28		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 27			0	MUX 26			0	MUX 25			0	MUX 24		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1256. Level1 Mux Selection 3 (L1SEL3)

62.5.1.1.5 Level1 Mux Selection 4 (L1SEL4)

Figure 1257 shows the format of the L1SEL4 register. The values for the fields of the L1SEL4 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x05 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 39			0	MUX 38			0	MUX 37			0	MUX 36		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 35			0	MUX 34			0	MUX 33			0	MUX 32		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1257. Level1 Mux Selection 4 (L1SEL4)

62.5.1.1.6 Level1 Mux Selection 5 (L1SEL5)

Figure 1258 shows the format of the L1SEL5 register. The values for the fields of the L1SEL5 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x06 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 47			0	MUX 46			0	MUX 45			0	MUX 44		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 43			0	MUX 42			0	MUX 41			0	MUX 40		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1258. Level1 Mux Selection 5 (L1SEL5)

62.5.1.1.7 Level1 Mux Selection 6 (L1SEL6)

Figure 1259 shows the format of the L1SEL6 register.The values for the fields of the L1SEL6 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x07 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 55			0	MUX 54			0	MUX 53			0	MUX 52		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 51			0	MUX 50			0	MUX 49			0	MUX 48		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1259. Level1 Mux Selection 6 (L1SEL6)

62.5.1.1.8 Level1 Mux Selection 7 (L1SEL7)

Figure 1260 shows the format of the L1SEL7 register. The values for the fields of the L1SEL7 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured. All eight inputs of MUX 63 are tied to logic high (1’b1) and output is always high.

Offset: 0x08 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 63			0	MUX 62			0	MUX 61			0	MUX 60		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 59			0	MUX 58			0	MUX 57			0	MUX 56		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1260. Level1 Mux Selection 7 (L1SEL7)

62.5.1.1.9 Input triggers level detection control 2 (ITLD2)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and therefore these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. *Figure 1261* shows the format of the ITLD2 register. All of the Mux fields in the ITLD2 register have the same description.

Offset: 0x0B Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Mux47		Mux46		Mux45		Mux44		Mux43		Mux42		Mux41		Mux40	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	Mux39		Mux38		Mux37		Mux36		Mux35		Mux34		Mux33		Mux32	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1261. Input triggers level detection control 2 (ITLD2)

The ITLD2 registers Mux fields are described in *Table 1269*.

Table 1269. ITLD2 register field descriptions

Field	Description
0:1 Mux*47*	Mux47 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
2:3 Mux*46*	Mux46 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1269. ITLD2 register field descriptions (continued)

Field	Description
4:5 Mux*45*	Mux45 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
6:7 Mux*44*	Mux44 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
8:9 Mux*43*	Mux43 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
10:11 Mux*42*	Mux42 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
12:13 Mux*41*	Mux41 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
14:15 Mux*40*	Mux40 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
16:17 Mux*39*	Mux39 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
18:19 Mux*38*	Mux38 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
20:21 Mux*37*	Mux37 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1269. ITLD2 register field descriptions (continued)

Field	Description
22:23 Mux*36*	Mux36 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
24:25 Mux*35*	Mux35 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
26:27 Mux*34*	Mux34 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
28:29 Mux*33*	Mux33 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
30:31 Mux*32*	Mux32 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

62.5.1.1.10 Input triggers level detection control 3 (ITLD3)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and so these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. *Figure 1262* shows the format of the ITLD3 register.All of the Mux fields in the ITLD3 register have the same description.

Offset: 0x0C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Mux63		Mux62		Mux61		Mux60		Mux59		Mux58		Mux57		Mux56	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	Mux55		Mux54		Mux53		Mux52		Mux51		Mux50		Mux49		Mux48	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1262. Input triggers level detection control 3 (ITLD3)

The ITLD3 registers Mux fields are described in *Table 1270*.

Table 1270. ITLD3 register field descriptions

Field	Description
0:1 Mux*63*	Mux63 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
2:3 Mux*62*	Mux62 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
4:5 Mux*61*	Mux61 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
6:7 Mux*60*	Mux60 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
8:9 Mux*59*	Mux59 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
10:11 Mux*58*	Mux58 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
12:13 Mux*57*	Mux57 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
14:15 Mux*56*	Mux56 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
16:17 Mux*55*	Mux55 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1270. ITLD3 register field descriptions (continued)

Field	Description
18:19 Mux*54*	Mux54 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
20:21 Mux*53*	Mux53 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
22:23 Mux*52*	Mux52 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
24:25 Mux*51*	Mux51 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
26:27 Mux*50*	Mux50 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
28:29 Mux*49*	Mux49 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
30:31 Mux*48*	Mux48 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

62.5.1.1.11 CPU2 processor exception vector prefix (C2PEVP)

The SPU captures the exception vector from CPU2 when the exception enable signal is asserted. The exception vector decoding is defined in *Table 1294*. The SPU compares this captured vector with the value in the C2PEVP register. *Figure 1263* shows the format of the C2PEVP register.

Offset: 0x0F Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	CPU2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1263. CPU2 processor exception vector prefix (C2PEVP)

The C2PEVP register is described in *Table 1271*.

Table 1271. C2PEVP register field descriptions

Field	Description
0:15 CPU2	CPU2 processor exception vector Program this field with one of the vector values from *Table 1294* to select the corresponding processor exception. **Note:** This field must use the byte swapped values in the CnPEVP register CPUn field value column in *Table 1294*.

62.5.1.1.12 CPU2 interrupt priority selection (C2PIS)

Figure 1264 shows the format of the C2PIS register. The SPU compares the input CPR vector from CPU2 with the value programmed to the C2PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x14 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CPU2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1264. CPU2 interrupt priority selection (C2PIS)

The C2PIS register is described in *Table 1272*.

Table 1272. C2PIS register field descriptions

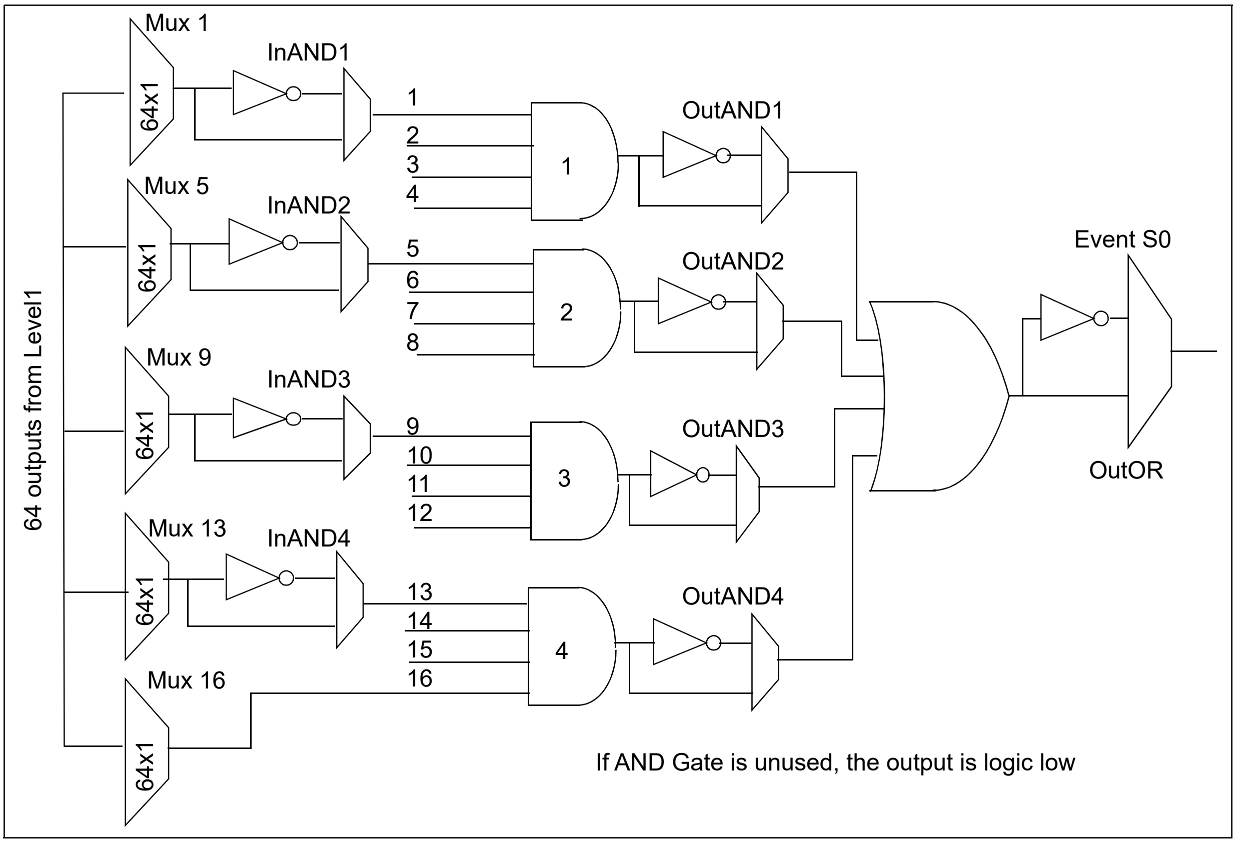
Field	Description
24:31 CPU2	CPU2 Priority Interrupt Selection

62.5.1.2 Level2 input Mux configurations registers

At level1, the set of input Mux control registers (described in *Section 62.5.1.1: Level1 input Mux configurations registers*) narrows down the large number of SPU watchpoints to a manageable set using sixty-four 8 × 1 multiplexers. At Level2, 16 inputs are selected for each SLU from the 64 level1 outputs using 16 64 × 1 multiplexers. Six bits per Mux select one input out of 64 (in fact 63) inputs. Each 32-bit register configures the AND gates of one state; therefore, eight registers are used to configure inputs for the 8 SLUs. All the 63 inputs are static for all the eight states. Any 16 inputs can be selected from the 63 input triggers for a state.

The detail selection is shown in the *Figure 1286* for State0. Similar logic is provided for States 1 to 7.

Figure 1265. Level2 Mux AND gates input pin routing for State0



Note: If any SLU AND gate is unused, then its output is logic low (1'b0). Thus, if all four inputs of a particular AND gate are at the default value 1'b1 (none of the inputs have been routed from the Level 2 mux to the inputs of AND gates), then the output of that AND gate is logic low. This provides protection to avoid the masking of other inputs to the OR gate.

62.5.1.2.1 Level2 Mux state n selection 0 (L2nSEL0)

Figure 1266 shows the format of the L2nSEL0 register where the state number, n = 0 to 7.

Offset: 0x17 +n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FirstANDInput4						0	0	FirstANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FirstANDInput2						0	0	FirstANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1266. Level2 Mux state n selection 0 (L2nSEL0)

The L2nSEL0 register fields are described in *Table 1273*.

Table 1273. L2nSEL0 register field descriptions

Field	Description
2:7 FirstANDInput4	First AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 FirstANDInput3	First AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

Table 1273. L2nSEL0 register field descriptions (continued)

Field	Description
18:23 FirstANDInput2	First AND Gate Input 2 Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 FirstANDInput1	First AND Gate Input 1 Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

62.5.1.2.2 Level2 Mux state n selection 1 (L2nSEL1)

Figure 1267 shows the format of the L2nSEL1 register where the state number, n = 0 to 7.

Offset: 0x18 +n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SecondANDInput4						0	0	SecondANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	SecondANDInput2						0	0	SecondANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1267. Level2 Mux state n selection 1 (L2nSEL1)

The L2nSEL1 register fields are described in *Table 1274*.

Table 1274. L2nSEL1 register field descriptions

Field	Description
2:7 SecondANDInput4	Second AND Gate Input 4 Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 SecondANDInput3	Second AND Gate Input 3 Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 SecondANDInput2	Second AND Gate Input 2 Selection 000001 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 SecondANDInput1	Second AND Gate Input 1Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

62.5.1.2.3 Level2 Mux state n selection 2 (L2nSEL2)

Figure 1268 shows the format of the L2nSEL2 register where the state number, n = 0 to 7.

Offset: 0x19 n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ThirdANDInput4						0	0	ThirdANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ThirdANDInput2						0	0	ThirdANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1268. Level2 Mux state n selection 2 (L2nSEL2)

The L2nSEL2 register fields are described in *Table 1275*.

Table 1275. L2nSEL2 register field descriptions

Field	Description
2:7 ThirdANDInput4	Third AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 ThirdANDInput3	Third AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 ThirdANDInput2	Third AND Gate Input 2 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 ThirdANDInput1	Third AND Gate Input 1 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

62.5.1.2.4 Level2 Mux state n selection 3 (L2nSEL3)

Figure 1269 shows the format of the L2nSEL3 register where the state number, n = 0 to 7.

Offset: 0x1A +n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FourthANDInput4						0	0	FourthANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FourthANDInput2						0	0	FourthANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1269. Level2 Mux state n selection 3 (L2nSEL3)

The L2nSEL3 register fields are described in *Table 1276*.

Table 1276. L2nSEL3 register field descriptions

Field	Description
2:7 FourthANDInput4	Fourth AND Gate Input 4 Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 FourthANDInput3	Fourth AND Gate Input 3 Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 FourthANDInput2	Fourth AND Gate Input 2 Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 FourthANDInput1	Fourth AND Gate Input 1 Selection 000000 No Input is selected (tie to 1’b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

62.5.1.3 Sequence unit registers

62.5.1.3.1 Input/output Inversion control state n (IOICn)

An input control register is provided for each event generated by the SLU. This register is programmed to specify how the input signals to each SLU are combined to generate the resulting event for that particular sequence. In total, there are 16 inputs for each state and only one (the first) of the inputs can be used with or without inversion. The input can be left unused if not required for that state by selecting 000000 (No input selected option) in the L2nSEL registers. The output of each AND gate has the optional inversion control as well. The output can be routed directly or with inversion from each AND gate to the input of OR gate. *Figure 1270* shows the format of the IOICn register where the state number, n = 0 to 7.

Offset: 0x37 n*0x1 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	OutOR	OutAND4	OutAND3	OutAND2	OutAND1	InAND4	InAND3	InAND2	InAND1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1270. Input/output Inversion control state n (IOICn)

The IOICn register fields are described in *Table 1277*.

Table 1277. IOICn register field descriptions

Field	Description
23 OutOR	Output OR GATE Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
24 OutAND4	Output of AND Gate 4 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
25 OutAND3	Output AND Gate 3 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
26 OutAND2	Output AND Gate 2 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
27 OutAND1	Output AND Gate 1 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event

Table 1277. IOICn register field descriptions (continued)

Field	Description
28 InAND4	Input 1 AND Gate 4 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
29 InAND3	Input 1 AND Gate 3 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
30 InAND2	Input 1 AND Gate 2 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
31 InAND1	Input 1 AND Gate 1 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event

62.5.1.3.2 Sequence control (SCTRL)

This configuration register is provided to enable the sequence and to select the states that are part of a particular sequence. There can be at most four active sequences. Each sequence can have one state or a maximum of eight states. However, each state can only be used in one unique sequence. In other words, if one state has been selected for a sequence, then this state cannot be used for any other defined sequence. Each sequence has a start state and end state. *Figure 1271* shows the format of the SCTRL register.

Offset: 0x3F Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S4ST2			S4ST1			S4ST0	0	S3ST2			S3ST1			S3ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	S2ST2			S2ST1			S2ST0	0	S1ST2			S1ST1			S1ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1271. Sequence control (SCTRL)

The SCTRL register fields are described in *Table 1278*.

Table 1278. SCTRL register field descriptions

Field	Description
1:3 S4ST2	Sequence 4 End State 000 State 0 is the Stop state for sequence 4 001 State 1 is the Stop state for sequence 4 010 State 2 is the Stop state for sequence 4 011 State 3 is the Stop state for sequence 4 100 State 4 is the Stop state for sequence 4 101 State 5 is the Stop state for sequence 4 110 State 6 is the Stop state for sequence 4 111 State 7 is the Stop state for sequence 4
4:6 S4ST1	Sequence 4 Start State(1) 000 State 0 is the Start state for sequence 4 001 State 1 is the Start state for sequence 4 010 State 2 is the Start state for sequence 4 011 State 3 is the Start state for sequence 4 100 State 4 is the Start state for sequence 4 101 State 5 is the Start state for sequence 4 110 State 6 is the Start state for sequence 4 111 State 7 is the Start state for sequence 4
7 S4ST0	Sequence 4 Enable/Disable(2) Once enabled, this bit is disabled at the end of the Sequence 4. 0 Sequence 4 is disabled 1 Sequence 4 is enabled
9:11 S3ST2	Sequence 3 End State 000 State 0 is the Stop state for sequence 3 001 State 1 is the Stop state for sequence 3 010 State 2 is the Stop state for sequence 3 011 State 3 is the Stop state for sequence 3 100 State 4 is the Stop state for sequence 3 101 State 5 is the Stop state for sequence 3 110 State 6 is the Stop state for sequence 3 111 State 7 is the Stop state for sequence 3
12:14 S3ST1	Sequence 3 Start State(1) 000 State 0 is the Start state for sequence 3 001 State 1 is the Start state for sequence 3 010 State 2 is the Start state for sequence 3 011 State 3 is the Start state for sequence 3 100 State 4 is the Start state for sequence 3 101 State 5 is the Start state for sequence 3 110 State 6 is the Start state for sequence 3 111 State 7 is the Start state for sequence 3
15 S3ST0	Sequence 3 Enable/Disable(2) Once enabled, this bit is disabled at the end of the Sequence 3. 0 Sequence 3 is disabled 1 Sequence 3 is enabled

Table 1278. SCTRL register field descriptions (continued)

Field	Description
17:19 S2ST2	Sequence 2 End State 000 State 0 is the Stop state for sequence 2 001 State 1 is the Stop state for sequence 2 010 State 2 is the Stop state for sequence 2 011 State 3 is the Stop state for sequence 2 100 State 4 is the Stop state for sequence 2 101 State 5 is the Stop state for sequence 2 110 State 6 is the Stop state for sequence 2 111 State 7 is the Stop state for sequence 2
20:22 S2ST1	Sequence 2 Start State(1) 000 State 0 is the Start state for sequence 2 001 State 1 is the Start state for sequence 2 010 State 2 is the Start state for sequence 2 011 State 3 is the Start state for sequence 2 100 State 4 is the Start state for sequence 2 101 State 5 is the Start state for sequence 2 110 State 6 is the Start state for sequence 2 111 State 7 is the Start state for sequence 2
23 S2ST0	Sequence2 Enable/Disable(2) Once enabled, this bit is disabled at the end of the Sequence 2. 0 Sequence 2 is disabled 1 Sequence 2 is enabled
25:27 S1ST2	Sequence 1 End State 000 State 0 is the Stop state for sequence 1 001 State 1 is the Stop state for sequence 1 010 State 2 is the Stop state for sequence 1 011 State 3 is the Stop state for sequence 1 100 State 4 is the Stop state for sequence 1 101 State 5 is the Stop state for sequence 1 110 State 6 is the Stop state for sequence 1 111 State 7 is the Stop state for sequence 1
28:30 S1ST1	Sequence 1 Start State 000 State 0 (Always)
31 S1ST0	Sequence1 Enable/Disable(2) Once enabled, this bit is disabled at the end of the Sequence 1. 0 Sequence 1 is disabled 1 Sequence 1 is enabled

1. Ideally, the start state for a sequence should be the state immediately after the stop state for the previous sequence. For example, if sequence 1 starts at state 0 and ends at state3, sequence 2 should start at state4.
2. Multiple active sequences such as Sequence 2, Sequence 3 and Sequence 4 must be programmed with Div4 option only.

62.5.1.3.3 State n goto control (SnGC)

SLU states are linked to each other based on if-then-else conditions. There are two conditions: true and false. A SLU state can remain in the same state or jump to any other state. The state goto control register is programmed to specify the GoTo a particular state from current state.

Figure 1272 shows the format of the SnGC register where the state number, n = 0 to 7.

Offset: 0x40 +n*0x1 (n= 0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	NF			FE	NT			TE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1272. State n goto control (SnGC)

The SnGC register fields are described in *Table 1279*.

Table 1279. SnGC register field descriptions

Field	Description
24:26 NF	Next State on False Condition Defines the Goto next state in case of false condition from the present state. States may remain in the same state on a false condition if the same state is chosen as the next state. 000 Goto State 0 001 Goto State 1 010 Goto State 2 011 Goto State 3 100 Goto State 4 101 Goto State 5 110 Goto State 6 111 Goto State 7
27 FE	GOTO Control of State Under False Condition Defines the Goto control of a state under the false condition. If this bit is enabled, then the state moves to the next state based on bits[7:5]. If disabled, then this is the is end of a sequence when the sequence comes to this state. Actions associated with states under false condition are executed. Disabling this bit end the sequence under a false condition. 0 Disable Goto control 1 Enable Goto control

Table 1279. SnGC register field descriptions (continued)

Field	Description
28:30 NT	Next State on True Condition Defines the Goto next state in case of true condition from the present state. States may remain in the same state on a true condition if the same state is chosen as the next state. 000 Goto State 0 001 Goto State 1 010 Goto State 2 011 Goto State 3 100 Goto State 4 101 Goto State 5 110 Goto State 6 111 Goto State 7
31 TE	GoTo Control of State Under True Condition Defines the Goto control of a state under the true condition. If this bit is enabled, then the State moves to the next state based on bits[3:1]. If disabled, then this is the end of a sequence when the sequence comes to this state. Actions associated with states under true condition are executed. Disabling this bit ends the sequence under a true condition. 0 Disable Goto control 1 Enable Goto control

62.5.1.4 SLU status register (SS)

The SLU status register indicates the states that have been hit/encountered in any of the active sequences. *Figure 1273* shows the format of the SS register.

Offset: 0x48 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	EM2	0	0	SAS4			SAS3
W										w1c			w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAS3		SAS2			SAS1			SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W	w1c		w1c			w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1273. SLU status register (SS)

The SS register fields are described in *Table 1280*.

Table 1280. SS register field descriptions

Field	Description
6 IM2	Interrupt Match 2 This bit indicates the status of the match value on interrupt priority register of CPU2. 0 No match 1 Match occurred
9 EM2	Exception Match 2 This bit indicates the status of the processor exception vector match value of CPU2. 0 No match 1 Match occurred
12:14 SAS4	Sequence 4 Active State Two consecutive read cycles are required to know the active state.These bits capture the active state of sequence 4 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
15:17 SAS3	Sequence 3 Active State Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 3 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 state5 is the active state 110 State6 is the active state 111 State7 is the active state
18:20 SAS2	Sequence 2 Active State Two consecutive read cycles are required to know the active state.These bits capture the active state of sequence 2 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state

Table 1280. SS register field descriptions (continued)

Field	Description
21:23 SAS1	Sequence 1 Active State Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 1 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
24 SH7	State Hit 7 This bit is set when State7 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
25 SH6	State Hit 6 This bit is set when State6 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
26 SH5	State Hit 5 This bit is set when State5 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
27 SH4	State Hit 4 This bit is set when State4 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
28 SH3	State Hit 3 This bit is set when State3 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
29 SH2	State Hit 2 This bit is set when State2 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
30 SH1	State Hit 1 This bit is set when State1 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
31 SH0	State Hit 0 This bit is set when State0 is hit/encountered in any of the active sequences 0 State not hit 1 State hit

62.5.1.5 SPU enable register (SE)

The SPU enable/disable register is used to enable and disable the SPU. If disabled, all inputs are masked. *Figure 1274* shows the format of the SE register.

Offset: 0x49 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV	SED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1274. SPU enable register (SE)

The SE register fields are described in *Table 1281*.

Table 1281. SE register field descriptions

Field	Description
30 DIV	Div Option Select Div4 and Div1 programmable options based on number of active sequences. 0 Use Div4 option, when multiple up to four sequences are active or only one sequence is active which does not require the highest resolution 1 Use a Div1 option, when only one sequence (Sequence1) is active and highest resolution is required
31 SED	SPU Enable/Disable This bit should be set after programming all of the SPU configuration registers. 0 SPU is disabled 1 SPU is enabled

Note: Div option mapping with active sequences must be correctly programmed by the user. No hardware checks are implemented.

62.5.1.6 Sample and hold mechanism

SPU supports Div4/Div1 sample and state processing frequency (both are effectively the same). For the Div1 option, the sample and state processing frequency is the SPU clock frequency. Likewise, for the Div4 option, the sample and state processing frequency is 1/4th the SPU clock frequency.

The Div1 or Div4 is related to hold circuitry (among other circuits as well), but the sample circuitry is always Div1. Therefore, if Div4 is selected for hold circuitry, then four samples are taken and then latched into a hold circuitry (for the next four Div1 clocks, synchronized for all inputs).

This sample and hold mechanism is applicable for external signals only, but counters logic process differently. These external signals may be considered as selected conditions for any sequence, and for any of the states of any sequence. There is no difference based on state or sequence, as all inputs are treated the same regardless.

62.5.1.6.1 Functional limitations

If there are multiple true conditions in a Div4 period, only one condition is registered. Never are there any missed true conditions, except for extra true conditions in a Div4 period. Therefore, the user should select the resolution of SPU based on number of active sequences.

- If the signal changes faster than the sample and state frequency, then the SPU would probably only detect one event per the sample period (for example, if there are multiple true conditions in a Div4 period, only one condition is registered).
- If the signal changes faster than the sample and state frequency, then the SPU would only be able to process one event per state processing frequency.

62.5.1.6.2 Assumption

When using Div1, SPU counter operations are not available for state evaluation in the immediately next clock cycle. Therefore, special handling must be done in state evaluation during this delayed effect to assure that it does not create undesirable results. Refer to counter workaround description in *Section 62.2*.

When using Div4, however, there are no delayed effects when using SPU counter operations. This is because subsequent state evaluations are done four clocks apart, instead of every clock for Div1 selection. Therefore, counter workaround description in *Section 62.2*, becomes redundant with Div4 option.

62.5.1.7 SLU action unit registers

The action unit of the SPU is responsible for generating the action triggers supported by various actions, listed in *Section 62.2*. The action triggers are mapped to the events by programming the action registers. Each event can be programmed to trigger interrupts, watchpoints, trace signals, counter stop/reset/increment and many other general actions. *Table 1282* shows the bit mapping for all the possible actions divided in three groups: client based, counter based, system related.

Table 1282. SPU action decoder

Group bit [11:10] ID	Group bit [11:10] Name	Client/counter bit [9:5] ID	Client/counter bit [9:5] Name	Action bit [4:1] ID	Action bit [4:1] Name	Action bit [4:1] Comment	Enable bit [0]
00	Client group	00000	Reserved	0000	—	—	0=Stop 1=Start
		00001	Reserved	—	—	Valid for Cores, AHB and trace group actions	
		00010	CPU2	0010			
		00011–00111	Reserved	0011	—		
		01000	Reserved	0100	—		
		01001–01111	Reserved	0101	PMC1—	Valid for the cores as PMC—	For PMCn 0= Stop 1= Start
		10000	NXMC1	0110	PMC2—		
		10001	NXMC2	0111	PMC3—		
		10010–10111	Reserved	1000	PMC4—		
		11000	Reserved	1001	—	—	0=Stop 1=Start
		11001–11110	Reserved	1010	—		
				1011	—		
				1100	—		
		11111	Trace group(1)	1101–1111	Reserved	Valid for CPU cores and AHB	
01	Counter group	00000	Counter0	0000	No action	—	Don't care
		00001	Counter1	0001	Start timer/Incr counter	Valid for all counters/timers	
		00010	Counter2	0010	Stop timer		
		…	…	0011	Reset counter		
		01111	Counter15	0100	Capture		
		10000–11111	Reserved	0101–1111	Reserved		

Table 1282. SPU action decoder (continued)

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
10	System group	00000	System action	0000	No action	—	Don’t care
		00001–11111	Reserved	0001	Halt1_dci_evti	—	
				0010	Halt2_dci_evti	—	
				0011	—	—	
				0100	—	—	
				0101	—	—	
				0110	—	—	
				0111	Evto1	—	
				1000	Evto2	—	
				1001	Reserved	—	
				1010	Core INTR	—	
				1011–1111	Reserved	—	—
11	Reserved						

1. The valid actions for Trace group are OTM trace, PTM trace, DTM trace or WTM trace. No other actions can be selected with Trace group. The group can be defined using the Trace group configuration registers.

62.5.1.7.1 State n true action 0 (SnTA0)

The SPU can be programmed to enable up to four different actions against a true condition of every state. All the possible actions and their encoding are listed in *Table 1282*. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1.

Figure 1275 shows the format of the SnTA0 register where the state number, n = 0 to 7.

Offset: 0x01 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION2											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1275. State n true action 0 (SnTA0)

The SnTA0 register fields are described in *Table 1283*.

Table 1283. SnTA0 register field descriptions

Field	Description
4:15 ACTION2	Action 2 to be performed for State n true condition The decoding of these bits is defined in *Table 1282*.
20:31 ACTION1	Action 1 to be performed for State n true condition The decoding of these bits is defined in *Table 1282*.

62.5.1.7.2 State n true action 1 (SnTA1)

The SPU can be programmed to enable up to four different actions against a true condition of every state. All the possible actions and their encoding are listed in *Table 1282*. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1. *Figure 1276* shows the format of the SnTA1 register where the state number, n = 0 to 7.

Offset: 0x02 + n*0x2 (n= 0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION4											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION3											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1276. State n true action 1 (SnTA1)

The SnTA1 register fields are described in *Table 1284*.

Table 1284. SnTA1 register field descriptions

Field	Description
4:15 ACTION4	Action 4 to be performed for State n true condition The decoding of these bits is defined in *Table 1282*.
20:31 ACTION3	Action 3 to be performed for State n true condition The decoding of these bits is defined in *Table 1282*.

62.5.1.7.3 State n false action 0 (SnFA0)

The SPU can be programmed to enable up to four different actions against a false condition of every state. All the possible actions and their encoding are listed in *Table 1282*. A total of 12 bits are needed to code every action. The first two actions for a false condition are defined in SnFA0 and the next two actions are defined in SnFA1. *Figure 1277* shows the format of the SnFA0 register where the state number, n = 0 to 7.

Offset: 0x11 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION2											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1277. State n false action 0 (SnFA0)

The SnFA0 register fields are described in *Table 1285*.

Table 1285. SnFA0 register field descriptions

Field	Description
4:15 ACTION2	Action 2 to be performed for State n false condition The decoding of these bits is defined in *Table 1282*.
20:31 ACTION1	Action 1 to be performed for State n false condition The decoding of these bits is defined in *Table 1282*.

62.5.1.7.4 State n false action 1 (SnFA1)

The SPU can be programmed to enable up to four different actions against a false condition of every state. All the possible actions and their encoding are listed in *Table 1282*. A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. *Figure 1278* shows the format of the SnFA1 register where the state number, n = 0 to 7.

Offset: 0x12 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION4											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION3											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1278. State n false action 1 (SnFA1)

The SnFA1 register fields are described in *Table 1286*.

Table 1286. SnFA1 register field descriptions

Field	Description
5:15 ACTION4	Action 4 to be performed for State n false condition The decoding of these bits is defined in *Table 1282*.
20:31 ACTION3	Action 3 to be performed for State n false condition The decoding of these bits is defined in *Table 1282*.

62.5.1.7.5 Trace group configuration registers (DTMTGC, PTMTGC, OTMTGC, WTMTGC)

There are four action trace groups and a configuration register for each trace group:

- DTM trace: clients are configured with the DTM trace group configuration (DTMTGC) register
- PTM trace: clients are configured with the PTM trace group configuration (PTMTGC) register
- OTM trace: clients are configured with the OTM trace group configuration (OTMTGC) register
- WTM trace: clients are configured with the WTM trace group configuration (WTMTGC) register

Each of these groups is specific to a trace. Any number of clients can be enabled for one trace using the action trace group.

Figure 1279 shows the format of the trace group configuration registers.

Offset: 0x21 (DTMTGC)
0x22 (PTMTGC)
0x23 (OTMTGC)
0x24 (WTMTGC)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AHB2	AHB1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU2	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1279. Trace group configuration register format (DTMTGC, PTMTGC, OTMTGC, WTMTGC)

The trace group configuration register fields are described in *Table 1287*.

Table 1287. DTMTGC, PTMTGC, OTMTGC, WTMTGC field descriptions

Field	Description
14 AHB2	0 Disable trace for AHB2 1 Enable trace for AHB2
15 AHB1	0 Disable trace for AHB1 1 Enable trace for AHB1
29 CPU2	0 Disable trace for CPU2 1 Enable trace for CPU2

62.5.1.7.6 Interrupt status (INTS)

The status of the external interrupt request to the processor core through the JTAGM is registered in the Interrupt status (INTS) register. This status bit is cleared when the acknowledge is received from the JTAGM. *Figure 1280* shows the format of the INTS register.

Offset: 0x25 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1280. Interrupt status (INTS)

The INTS register fields are described in *Table 1288*.

Table 1288. INTS register field descriptions

Field	Description
31 INT	1 Interrupt status bit is cleared from JTAGM

62.5.1.8 Counters control n registers (CCTRLn)

There are 16 configurable counters/timers in the SPU. Configuration control of these counters is defined in the counter control registers. *Figure 1281* shows the format of the CCTRLn registers where the counter number, n = 0 to 15.

Offset: 0x26 + n*0x1 (n = 0 to 15) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	SW_RESET	PRESCALING			0	COUNT_INCR	MODE	ENABLE
W								0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1281. Counters control n registers (CCTRLn)

The CCTRLn register fields are described in *Table 1289*.

Table 1289. CCTRLn register field descriptions

Field	Description
23	Reserved. This bit is for internal use only and must be set to 0.
24 SW_RESET	Reset Counter can be reset by software. 0 Software reset is disabled 1 Software reset
25:27 PRESCALING	Prescaling Divides the timer clock by a defined value. 000 No scaling, divide by 1 001 Divide by 4 010 Divide by 16 011 Divide by 32 100 Reserved 101 Reserved 110 Reserved 111 Reserved
29 COUNT_INCR	Count Increment 0 Increment the counter based on the actions generated from the sequences 1 Increment the counter based on the PMC input. The mapping of the PMC inputs with the counters is shown in *Figure 1293*.
30 MODE	Mode Select Select counter or timer operation. 0 Counter 1 Timer
31 ENABLE	Enable 0 Counter disabled 1 Counter enabled

62.5.1.9 Counters compare n registers (CCMPn)

A 32-bit comparator value can be programmed for each of the counters. This value is compared with the corresponding counter/timer to generate an event for the SPU. *Figure 1282* shows the format of the CCMPn registers where the counter number, n = 0 to 15.

Offset: 0x36 + n*0x1 (n = 0 to 15) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	COMPARE_VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	COMPARE_VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1282. Counters compare n registers (CCMPn)

The CCMPn register fields are described in *Table 1290*.

Table 1290. CCMPn register field descriptions

Field	Description
0:31 COMPARE_VAL	Compare value This value is compared with the corresponding counter/timer value to generate an event for the SPU.

62.5.1.10 Status registers

62.5.1.10.1 Counter compare status (CCOMS)

The counter compare status register shows the status of the comparisons for all the counters. The bits in this register can be cleared by writing a 1 to them. *Figure 1283* shows the format of the CCOMS register. All of the CC fields in the CCOMS register have the same description as shown in *Table 1291*.

Note: Counter compare values and other settings are not recommended to change dynamically, so w1c on CCOMS should be performed in either condition:

- *Counter is in disabled state (ENABLE bit of CCTRLn set to 0)*
- *SW-Reset bit (SW_RESET) is set*

Offset: 0x46 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1283. Counter compare status (CCOMS)

The CCOMS register fields are described in *Table 1291*.

Table 1291. CCOMS register field descriptions

Field	Description
16 CC15	Counter 15 Compare 0 Counter status is not true 1 Counter status is true
17 CC14	Counter 14 Compare 0 Counter status is not true 1 Counter status is true
18 CC13	Counter 13 Compare 0 Counter status is not true 1 Counter status is true
19 CC12	Counter 12 Compare 0 Counter status is not true 1 Counter status is true
20 CC11	Counter 11 Compare 0 Counter status is not true 1 Counter status is true
21 CC10	Counter 10 Compare 0 Counter status is not true 1 Counter status is true
22 CC9	Counter 9 Compare 0 Counter status is not true 1 Counter status is true
23 CC8	Counter 8 Compare 0 Counter status is not true 1 Counter status is true
24 CC7	Counter 7 Compare 0 Counter status is not true 1 Counter status is true

Table 1291. CCOMS register field descriptions (continued)

Field	Description
25 CC6	Counter 6 Compare 0 Counter status is not true 1 Counter status is true
26 CC5	Counter 5 Compare 0 Counter status is not true 1 Counter status is true
27 CC4	Counter 4 Compare 0 Counter status is not true 1 Counter status is true
28 CC3	Counter 3 Compare 0 Counter status is not true 1 Counter status is true
29 CC2	Counter 2 Compare 0 Counter status is not true 1 Counter status is true
30 CC1	Counter 1 Compare 0 Counter status is not true 1 Counter status is true
31 CC0	Counter 0 Compare 0 Counter status is not true 1 Counter status is true

62.5.1.10.2 Counter overflow status (COS)

Whenever a counter overflow occurs, the corresponding counter overflow status register bit is set. The bits in this register can be cleared by writing a 1 to them. *Figure 1284* shows the format of the COS register. All of the CO fields in the COS register have the same description as shown in *Table 1292*.

Offset: 0x47 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CO15	CO14	CO13	CO12	CO11	CO10	CO9	CO8	CO7	CO6	CO5	CO4	CO3	CO2	CO1	CO0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1284. Counter overflow status (COS)

The COS register fields are described in *Table 1292*.

Table 1292. COS register field descriptions

Field	Description
16 CO15	Counter 15 Overflow 0 Counter did not overflow 1 Counter overflow occurred
17 CO14	Counter 14 Overflow 0 Counter did not overflow 1 Counter overflow occurred
18 CO13	Counter 13 Overflow 0 Counter did not overflow 1 Counter overflow occurred
19 CO12	Counter 12 Overflow 0 Counter did not overflow 1 Counter overflow occurred
20 CO11	Counter 11 Overflow 0 Counter did not overflow 1 Counter overflow occurred
21 CO10	Counter10 Overflow 0 Counter did not overflow 1 Counter overflow occurred
22 CO9	Counter 9 Overflow 0 Counter did not overflow 1 Counter overflow occurred
23 CO8	Counter 8 Overflow 0 Counter did not overflow 1 Counter overflow occurred
24 CO7	Counter 7 Overflow 0 Counter did not overflow 1 Counter overflow occurred
25 CO6	Counter 6 Overflow 0 Counter did not overflow 1 Counter overflow occurred
26 CO5	Counter 5 Overflow 0 Counter did not overflow 1 Counter overflow occurred
27 CO4	Counter 4 Overflow 0 Counter did not overflow 1 Counter overflow occurred
28 CO3	Counter 3 Overflow 0 Counter did not overflow 1 Counter overflow occurred
29 CO2	Counter 2 Overflow 0 Counter did not overflow 1 Counter overflow occurred

Table 1292. COS register field descriptions (continued)

Field	Description
30 CO1	Counter 1 Overflow 0 Counter did not overflow 1 Counter overflow occurred
31 CO0	Counter 0 Overflow 0 Counter did not overflow 1 Counter overflow occurred

62.5.1.10.3 Counter capture status (CCAPS)

Whenever a counter value is captured, the corresponding capture status register bit for that counter is set. If the counter capture value causes a FIFO overflow, the captured value is lost and the status bit is not set. The bits in this register can be cleared by writing a 1 to them. *Figure 1285* shows the format of the CCAPS register.

Offset: 0x48 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1285. Counter capture status (CCAPS)

The CCAPS register fields are described in *Table 1293*.

Table 1293. CCAPS register field descriptions

Field	Description
16 CC15	Counter 15 Captured 0 Counter not captured 1 Counter captured
17 CC14	Counter 14 Captured 0 Counter not captured 1 Counter captured
18 CC13	Counter13 Captured 0 Counter not captured 1 Counter captured
19 CC12	Counter 12 Captured 0 Counter not captured 1 Counter captured

Table 1293. CCAPS register field descriptions (continued)

Field	Description
20 CC11	Counter 11 Captured 0 Counter not captured 1 Counter captured
21 CC10	Counter 10 Captured 0 Counter not captured 1 Counter captured
22 CC9	Counter 9 Captured 0 Counter not captured 1 Counter captured
23 CC8	Counter 8 Captured 0 Counter not captured 1 Counter captured
24 CC7	Counter 7 Captured 0 Counter not captured 1 Counter captured
25 CC6	Counter 6 Captured 0 Counter not captured 1 Counter captured
26 CC5	Counter 5 Captured 0 Counter not captured 1 Counter captured
27 CC4	Counter 4 Captured 0 Counter not captured 1 Counter captured
28 CC3	Counter 3 Captured 0 Counter not captured 1 Counter captured
29 CC2	Counter 2 Captured 0 Counter not captured 1 Counter captured
30 CC1	Counter 1 Captured 0 Counter not captured 1 Counter captured
31 CC0	Counter 0 Captured 0 Counter not captured 1 Counter captured

62.6 Functional description

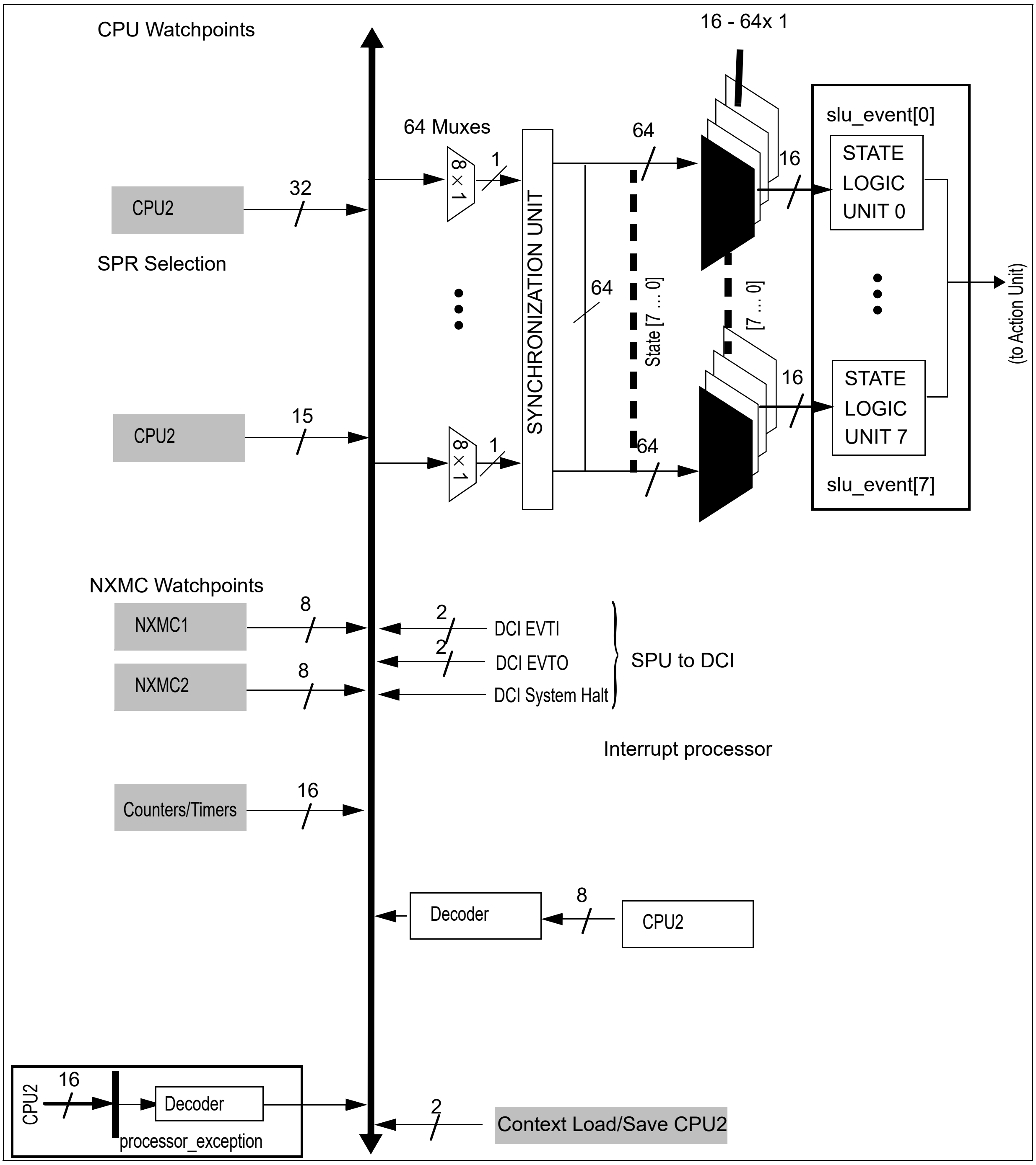
62.6.1 Input Mux unit

The SPU receives numerous inputs in the form of complex triggers from various clients to create a predefined sequence. One state logic can accommodate 16 inputs at one point of time. Therefore, an input Muxing unit becomes essential for reducing the debug and performance events to a manageable set of 16 inputs for each SLU. In addition to client input signals, there are certain performance events which are fed directly to the different counters inside the SPU.

The selection of 16 inputs for each state logic is done at two level of muxing both of which are shown in *Figure 1286*.

- Level 1 Muxing: At this level, there are 209 inputs. Out of these possible inputs, 63 inputs are selected. This selection is done using an 8:1 Mux. Therefore, sixty-four 8 × 1 muxes are required. The set of 63 watchpoints are passed to the second level of muxing. The selection mechanism is shown in *Figure 1286*.
- Level 2 Muxing: At this level, 16 inputs are selected out of the 63 possible inputs for each SLU. These 63 inputs consist of multiple set of 8 watchpoints from different clients, processor exception signals, interrupt from clients, and event IN/OUT signals. In addition, certain events from counters are used as feedback in the present state logic of each state.
  At level two there are 16 banks of 64:1 muxes for the selection of 16 inputs for one state logic. The inputs to these Muxes are expected to be somewhat unique. In general, care should be take to choose signal assignments that allow for key use cases to be achieved without encountering concurrency restrictions. Since there are eight such states, 16 × 8 banks of 64:1 mux are required. The selection mechanism is shown in *Figure 1286*.

Figure 1286. Input Mux unit block diagram



62.6.2 Processor exception vector encoding

Active-high input signals from the CPU indicate the particular exception being processed when the CPU exception enable signal is asserted. These signals can be used for debug triggering mechanisms. The SPU captures the CPU exception vector whenever the CPU exception enable signal is asserted and the SPU uses the exception vector as an input condition as shown in *Table 1294*. The CPU exception vector values from the CPU must be byte swapped before they are programmed to the CPUn field of the CnPEVP (n = 0) registers as shown in *Table 1294*.

Table 1294. Processor exception processing conditions

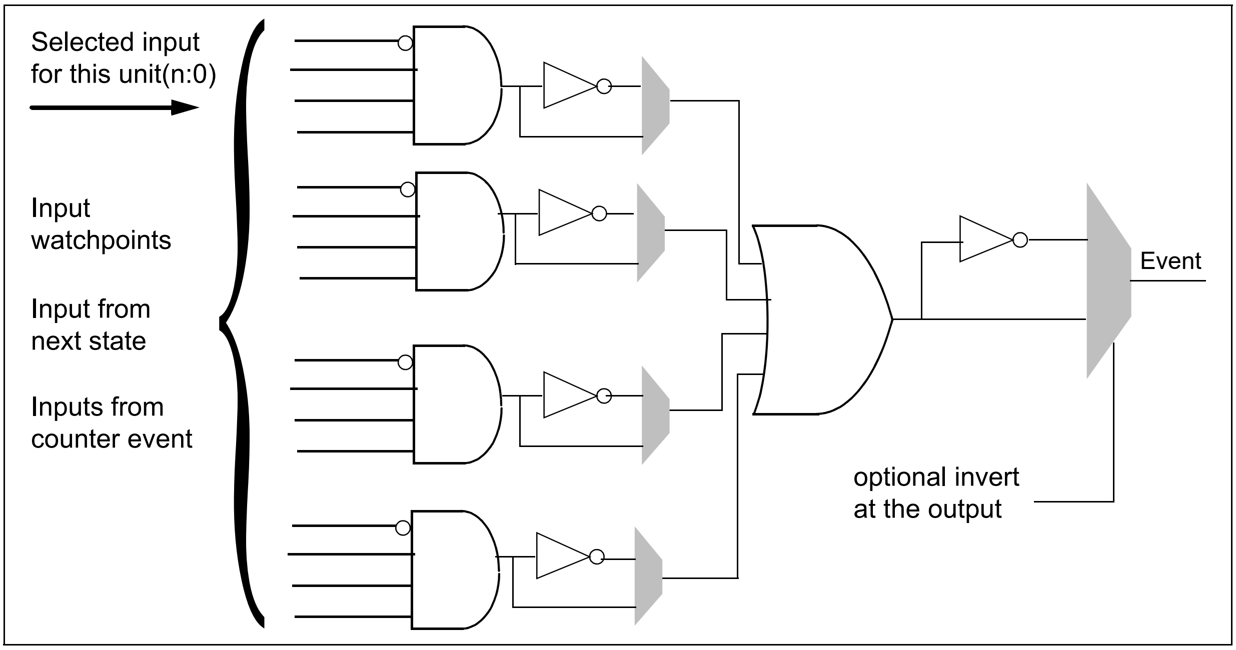
CPU exception vector	CnPEVP register CPUn field value	Exception condition
0x0000	0x0000	Critical Input: autovectored
0x0010	0x1000	Machine check
0x0020	0x2000	Data storage
0x0030	0x3000	Instruction storage
0x0040	0x4000	External input: autovectored
0x0050	0x5000	Alignment
0x0060	0x6000	Program trap
0x0061	0x6100	Program illegal
0x0062	0x6200	Program privileged
0x0070	0x7000	Performance monitor
0x0080	0x8000	System call
0x0090	0x9000	Debug
0x00A0	0xA000	EFPU data exception
0x00B0	0xB000	EFPU round exception
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	External input: vectored
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	Critical input: vectored

62.6.3 State logic unit (SLU)

Complex triggers and system performance monitor functions are implemented in the SLU. The SPU creates debug events based upon states in a sequence. A sequence can consist of a single state, or any number of states up to 8. Each state consists of combinational logic allowing AND/OR operations on inputs from the trigger source unit. Single or multiple (up to 8) actions can be triggered by a state machine.

A logic diagram of one SLU state is shown in *Figure 1287*.

Figure 1287. SLU block diagram



Each state consists of AND gates, the outputs of which are ORed together.

- Each AND gate can have up to 4 inputs, selected from the trigger source unit by the user.
- The user has an option to invert one input signal into the AND gate.
- The OR gate has a user selectable option to invert the output.
- The output of the OR gate is used as the trigger to move to the next state in the sequence.

The SLUs are responsible for performing operations on the selected input events to produce derived events. Each SLU has a control register that manages how the various inputs are combined to produce a derived event.

An event is triggered from each state based on all the possible selected inputs for each state. This event is used to trigger an Action.

Note: With only one optional inversion at the input of each AND gate and optional inversion at the output of the OR gate, many combinations are not possible.

62.6.4 Sequence formation

Event sequencing can be accomplished with the if-then-else operations. The next event can be fed back as a necessary condition for the current event. For example, event5 can be designated as a necessary condition for event4 in case test condition fails in event5. Event5 can be in turn designated as necessary for event6 in case test condition passes, and so on. In this way, events can be chained together to create a sequence.

To create complex triggers, a configurable state machine is implemented. This allows the user to join states together with if-then-else operations to create a sequence. The SPU supports up to four simultaneous sequences, however only eight states are supported regardless of the number of active sequences, and each state can only be used in one unique sequence.

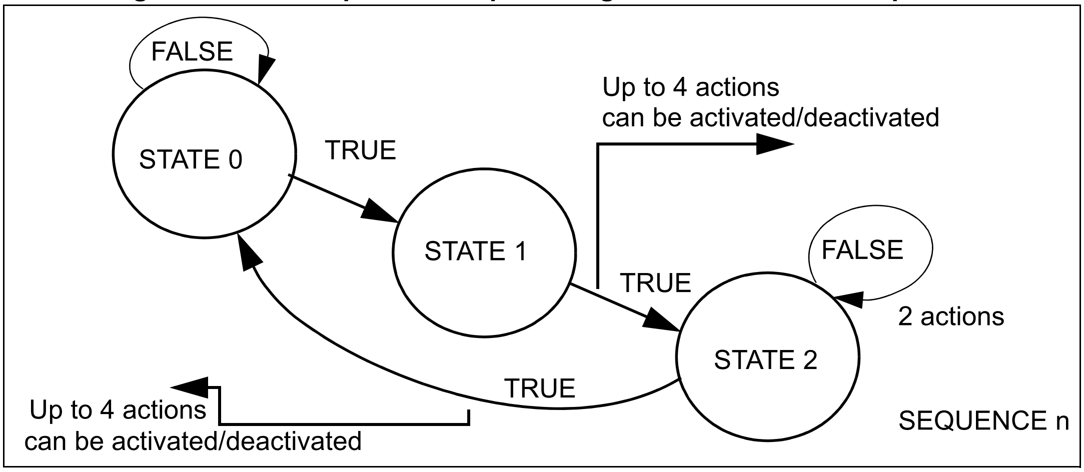
For example, the following state configurations are possible:

- 1 active sequence, with 8 states
- 2 active sequences, each with 4 states
- 2 active sequences, the first with 3 states and the second with 5 states
- 4 active sequences, each with 2 states
- 4 active sequences, each with 1 state

Each sequence has the following options which are shown in *Figure 1288*:

- The ability to optionally trigger one or more actions based on a true condition from any state in the sequence.
- The ability to optionally trigger one or more actions based on a false condition from any state in the sequence.
- Each state in a sequence has the ability to route to another state on a true condition from the state logic, and the ability to route to a different state based on a false condition from the state logic.

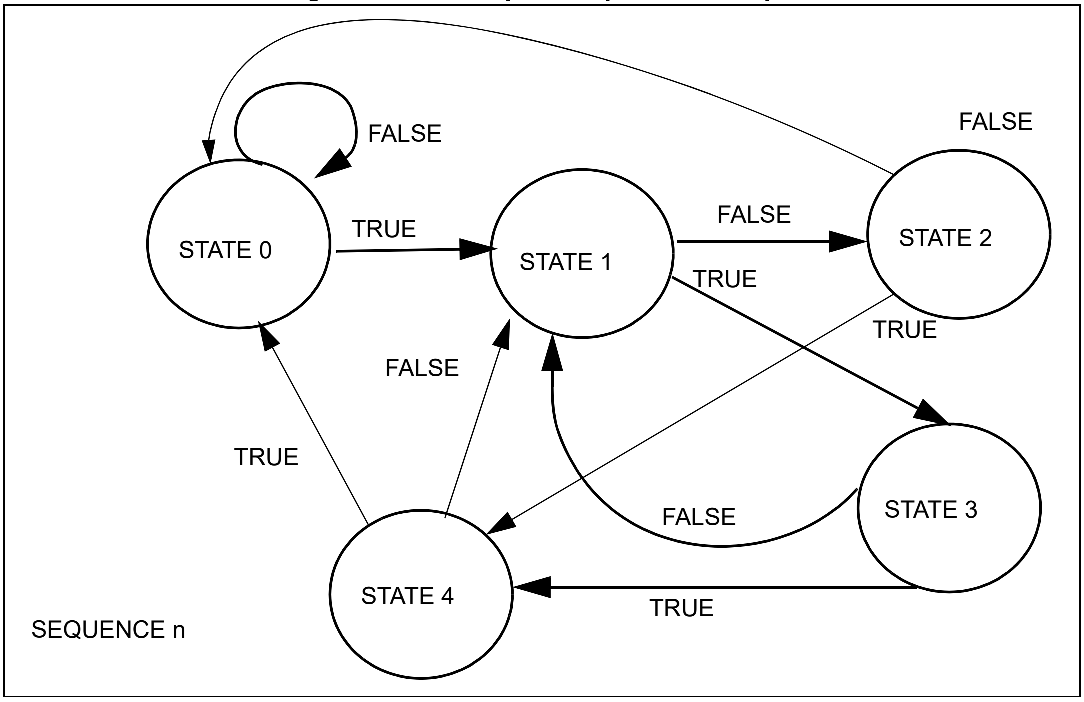
Figure 1288. Example of a simple configurable if-then-else sequence



62.6.4.1 Complex sequence example

A user can create a complex sequence using five states as shown in the state transition diagram in *Figure 1289*.

Figure 1289. Complex sequence example



62.6.4.2 Typical sequence example

For this example, the user would like to create a timestamp in the trace stream each time a function is entered and stop the trace signal on the 12th occurrence of the function. This use case can be implemented as follows:

- State 0 takes an instruction address compare as its input. On a match of the instruction address to a specific function, called function_x, the sequence moves to State 1. Upon this action, State0 is configured to insert a timestamp into the trace stream and also increment an SPU counter, say counter[0].
- The sequence is now in State 1. State 1 takes an input from counter[0]. Counter[0] has been pre-configured to create a match event when the counter value is equal to 12.
  - If the value of counter[0] is less than 12, then the sequence moves back to State0.
  - If the value of couner[0] is equal to 12, then the sequence completes and an action from State1 disables the trace signal.

62.6.4.3 Sequence status register description

A status register providing feedback on each sequence is provided to allow debug tools to identify the source of a debug event from the SPU. The status register consists of two field types:

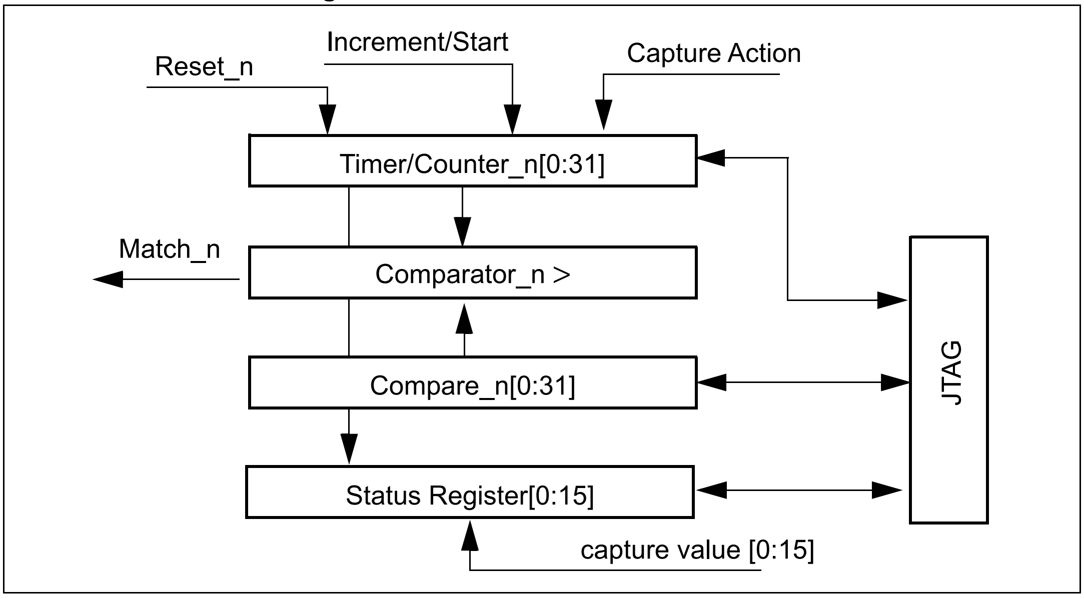
- States: each state has a sticky bit that is set upon a true match of the state logic. This sticky bit is cleared upon a write to this register
- Sequence status: each sequence has a field indicating the active state of the state machine. This is a read only field

62.6.5 Performance counters/timers unit

The 32-bit SPU counters may be used to delay events, count distances, count latencies or count events as shown in *Figure 1290*. Each counter can be configured as either a counter or a timer via the configuration register. The basic counting operation is straightforward: once enabled (default is disabled mode), the counter increments using the platform clock whenever the selected input is driven to a logic 1.

Access (via the JTAG interface) to each of the counter/timer registers allows the configuration of the timer/counter, the compare registers and global SPU timer/counter status registers. These registers are accessible via the JTAG interface even when the SPU is in disabled mode.

Figure 1290. Counters/timers overview



62.6.5.1 Counter events and actions

Each counter is capable of detecting one event condition, such as compare value match. The compare value match is used for triggering actions in the SPU. Each counter can be reset in response to a reset action generated from the SPU. Capture registers are available for capturing the counter state in response to a SPU trigger condition. This capture information is updated in the status register, which is accessible to users via JTAG interface. These status registers are type write 1 to clear (W1C).

There are two classes of signals, actions and events:

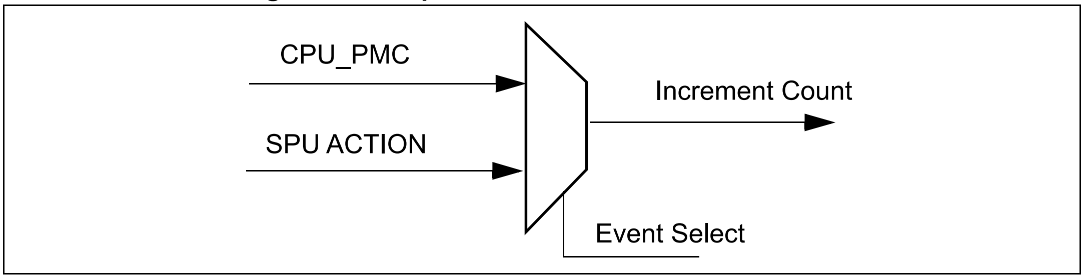
- Actions: each timer/counter has an interface from the action unit to start/increment and reset as appropriate. The start/increment signal is a single signal, with the operation selected in the SPU configuration registers. The generation of the appropriate count/reset signal is generated in the SPU.

  On an action to insert the timer/counter value into the trace stream, the value stored in the timer/counter is buffered inside the SPU and routed to the appropriate SoC block to insert the value into the trace stream one by one.
- Event: on a match event where a timer/counter value matches the value of the respective compare register, a signal is sent to the trigger source unit to indicate this.

62.6.5.2 Counter input selection at SPU level

Counter increment inputs derive from the SPU action or direct from the CPU performance measurement counters (CPU_PMC), as shown in *Figure 1291*. These inputs include signals from the primary SPU muxes or SPU events of counters.

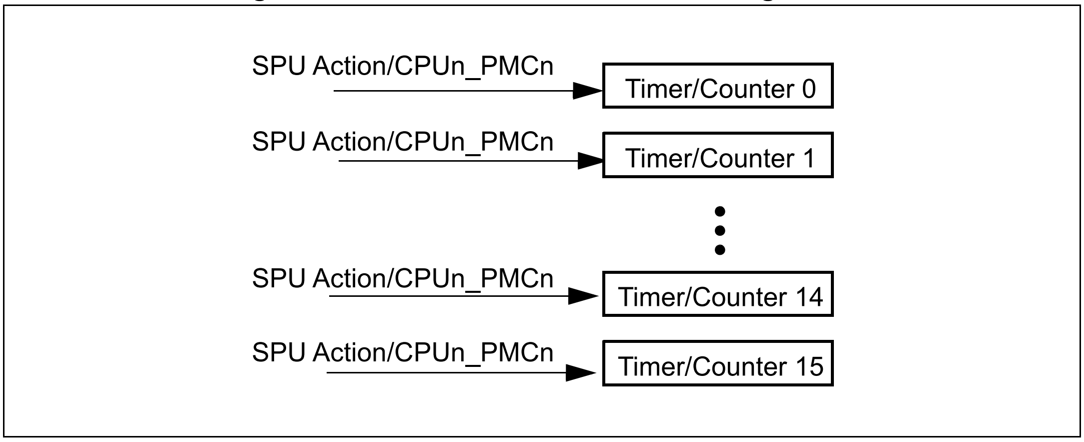
Figure 1291. nput selection for count increment



62.6.5.3 Counters increment assignment

When configured as a counter, the counter has an increment signal. This can optionally be routed from the SPU action unit, or directly from a CPU performance counter signal. The assignment of which signal is used to increment a counter is handled in the SPU, and is configurable by user. The assignment of increment signals to counters is shown in *Figure 1292*.

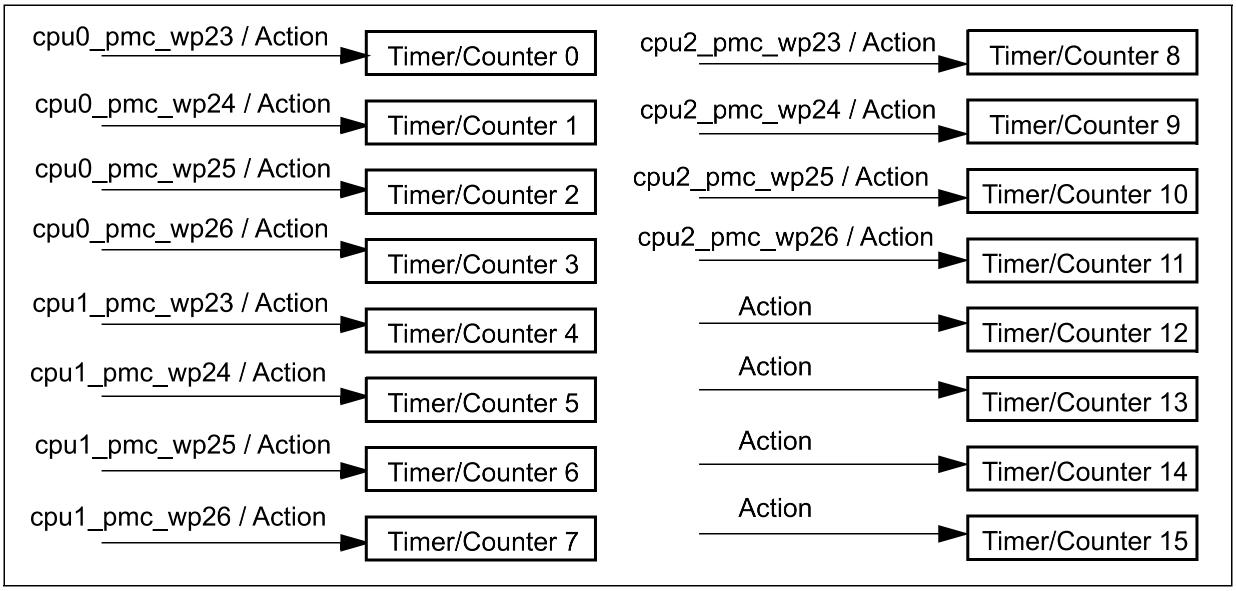
Figure 1292. SPU counters increment assignment



62.6.5.4 CPU PMC mapping to counters

Each of the CPUs in the system has four performance measurement counters (PMC). The outputs (overflow signals) from these counters (of CPUn) are routed to the SPU debug unit. The counters inside the SPU can be configured to count the pulses from the PMCs. There is a one-to-one mapping between the PMC inputs and the counters that they trigger. Their mapping is shown in the *Figure 1293*.

Figure 1293. CPU PMC mapping inside the SPU unit



62.6.6 Action unit

The action unit block of the SPU is responsible for the generating the action triggers supported by various actions. A state has true and false (if-else-than) condition. Four simultaneous actions can be generated under true/false condition of a particular state. The list of the actions is given in the *Section 62.2*. The actions triggers are mapped to the events by programming the set of SPU group action control for events registers. Each event can be programmed to trigger interrupts, timestamp watch points, trace signals, PMCs, start/reset/increment counters and many other general actions.

62.6.7 Optional trace group

User defined trace group can be defined through the configuration register. Configuration registers, defined in *Table 1287*, are available where the user can group the different traces from same or different clients.

62.6.8 Format for CKSRC & CKDATA Values

The format of the CKSRC is defined for the SPU Peripheral (as In-Circuit trace client):

Table 1295. CKSRC format

ICT CLIENTS	CKSRC Value
SPU Counter Trace Client	b00_0000
Reserved	b00_0001–b11_1111

The format for the CKDATA field for the SPU Counter trace client is:

Table 1296. CKDATA format

CKDATA	
CKDATA[CTR_Index]	CKDATA[CTR_VALUE]
4 bits	32-bits

63 JTAG Master (JTAGM)

63.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

63.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface or from the LFAST through the parallel interface. The JTAGM has the capability to differentiate between data from software and LFAST. It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI and transfers this data to the LFAST through another parallel interface. The JTAGM has the following registers:

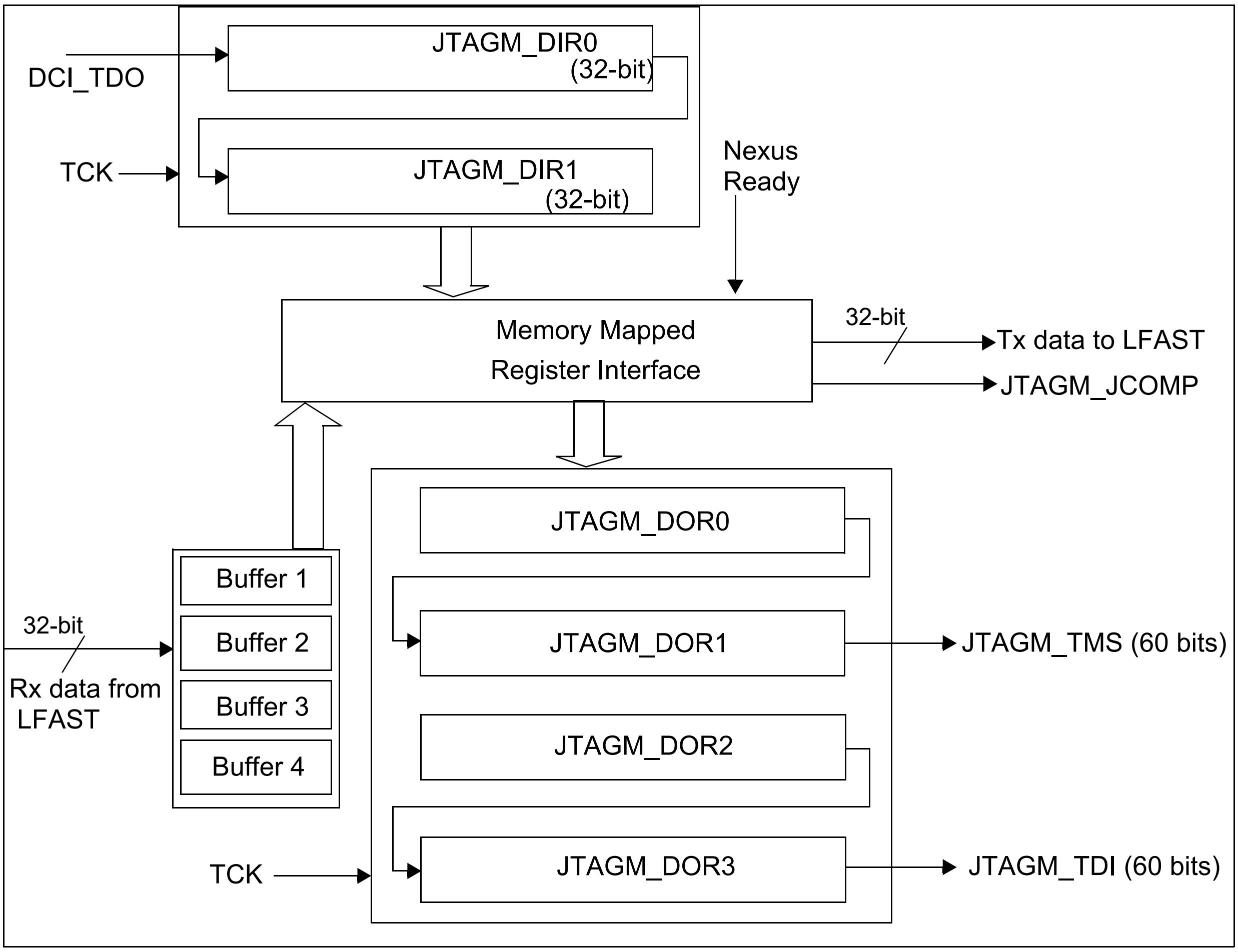
- Configuration register
- Status register
- Four data output registers
- RXCRC receive register
- Two data input registers

All these registers, described in detail in *Section 63.4*, are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

Figure 1294 shows the block diagram of the JTAGM.

Figure 1294. JTAGM block diagram



63.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG
- Receives/provides JTAG data from/to the LFAST

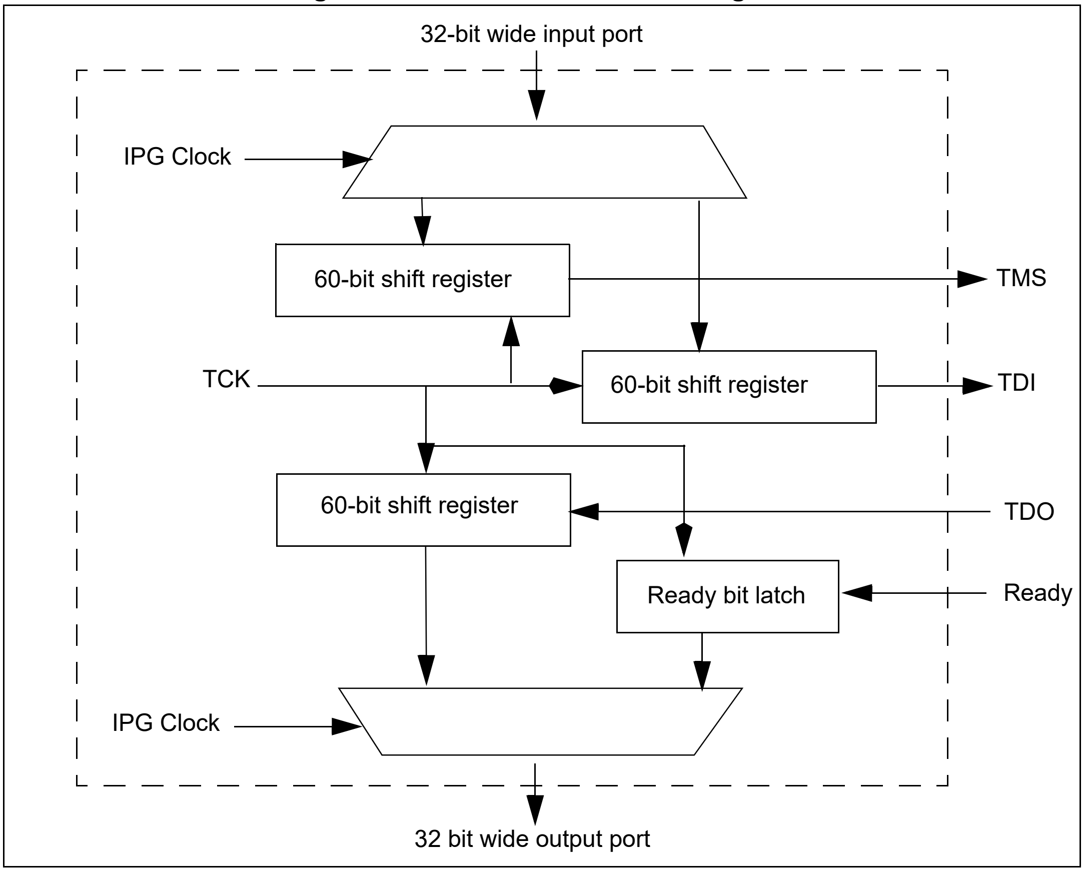
63.2 Functional description

63.2.1 Module architecture

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

Figure 1295 shows the flow diagram of the LFAST to DCI connection.

Figure 1295. LFAST-to-DCI flow diagram



63.2.2 JTAGM JCOMP usage in LFAST

JTAGM is used to drive JTAG sequences in software mode and in LFAST mode. The JTAGM_MCR[jtagm_JCOMP] bit is provided to reset all TAP controllers by generating a JCOMP signal to JTAGC.

The guidelines to be considered for the usage of JTAGM_MCR[jtagm_JCOMP]:

- Do not clear the JTAM_MCR[jtagm_JCOMP] bit while in LFAST mode as this operation is not guaranteed
- If for some reasons during the LFAST mode there is a loss of communication, one of the following should be done (instead of clearing the jtagm_JCOMP bit):
  - Generate the escape sequence and then, once the device is back in JTAG mode, generate a test-logic-reset state via the TCK and TMS signals. This resets all TAP controllers but does not affect the client debug configuration
  - Generate the escape sequence and then, once the device is back in JTAG mode, force the JCOMP pin low. This resets all TAP controllers as well as the client debug configuration

Note: The escape sequence is generated by forcing LFAST LVDS ports RX+ and RX- to zero.

63.2.3 JTAGM data error detection

To improve the integrity of the JTAG operation, data passed to and from the JTAGM includes an 8-bit CRC only in LFAST mode. The data sent to the JTAGM consists of 60 bits of TMS data, 60 bits of TDI and an 8-bit CRC. The JTAGM recalculates the 8-bit CRC for the first 120 bits of data and compares the result with the received CRC. If the two match, the data is assumed to be valid and the JTAGM converts this data into a valid JTAG frame.

If the CRCs do not match, the module sets the CRC error status bit (CRC_err) in the JTAGM status register (JTAGM_SR) and sends a standard 96-bit response frame to the LFAST channel B, with all 60 data bits set to 0. The CRC error status bit is automatically cleared as soon as the response frame has been sent to the LFAST.

Data sent from the JTAGM to the LFAST also includes an 8-bit CRC. The data sent from the JTAGM is in the form of a 96-bit frame consisting of 60 bits of TDO data + 4 bits of zero-padding + 32 bits of the status register. The 8-bit CRC value is calculated for the first 88 bits of the 96-bit frame and inserted into the least significant 8 bits of the status register; which is also the last 8 bits of the 96-bit frame. The tool is responsible for checking the CRC of the received data and taking corrective action.

63.2.4 JTAGM message tagging

The JTAGM, in conjunction with the LFAST, is pipelined to allow several JTAGM messages to be sent from the tool before any response is received back. This allows better utilization of the LFAST bandwidth. To enable the tool to associate individual messages with the correct response, the messages are tagged. The tool sends data messages on LFAST channels E, F, G and H in sequence. The JTAGM module returns responses on the same channel they were received. For example, if the tool sends a data command on channel E, JTAGM returns the response on channel E; if the tool sends a data command on channel F, the JTAGM returns the response on channel F; and so on. This way, the tool can ensure a response was generated by a specific command.

Configuration commands to write/read the configuration and status registers within the JTAGM are sent on channel A and response data is also sent back on channel A.

63.2.5 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available. In conventional JTAG mode, the status of this ready bit is also provided back to the tool via a dedicated pin to allow the tool to know data is ready and the JTAG read register can be accessed. With the LFAST interface providing pipelined data to the JTAGM, this technique is not available. Two interlocked mechanisms are provided within the JTAGM that use this ready bit to allow optimized data reads without ready bit errors.

63.2.5.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

63.2.5.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

63.2.6 $\overline{\text{EVTO}}$ and $\overline{\text{EVTI}}$ signals

The standard Nexus EVTI (Event In) and EVTO (Event Out) signals are optionally present on dedicated package pins and are used by an external tool in conjunction with the JTAG interface to pass information, events and triggers between the tool and the MCU.

The JTAGM as well as providing a higher speed alternative to the standard JTAG interface, also allows control and status of the EVTI and EVTO signals to be embedded into the serial protocol. The EVTI0 and EVTI1 signals can be asserted by writing control bits in the Module Configuration Register (JTAGM_MCR). The JTAGM can be configured by bits in the Module Configuration Register (JTAGM_MCR) to respond to a rising/falling/both edge transition of either EVTO0 or EVTO1. The response is either an unsolicited serial message to the tool when in LFAST mode (DTM = 0); or a CPU interrupt when in software mode (DTM = 1).

The unsolicited LFAST message is sent on LFAST channel B to allow the tool to distinguish the message from normal JTAG response data (channels E, F, G & H) and Config Command Responses (channel A). The CPU interrupt is gated by an Interrupt enable bit, in MCR and reported by Interrupt Status flags in the Status Register (JTAGM_SR). The status bits stay set until cleared by writing to the clear bits in the JTAG_SR.

Asserting the evti0_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. Asserting the evti1_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. The DTM bit in the JTAGM_MCR register can put the DCI in LFAST or SW mode.

Note: EVTI generation and EVTO detection bits of JTAGM can be used by debugger to detect EVTI assertion and to generate EVTO. This method is useful when EVTI/O pins are not available on SoC PADs.

63.2.7 JTAGM configuration and status monitoring

The JTAGM module requires some level of configuration and status monitoring. The JTAGM includes a 32-bit configuration register (JTAGM_MCR) and a 32-bit status register (JTAGM_SR, which includes the ready bit mentioned in *Section 63.2.5.1: Status register* ready bit). The configuration register is accessed via the parallel input port by a 128-bit payload arriving on channel A. The 128-bit payload is made from the following:

- 32-bit mask of which bits within the configuration register should be written
- 32-bit value to be written to the configuration register
- 24-bit mask of which bits within the status register should be written
- 8 bits of zero-padding
- 24 bits of data to be written to the status register
- 8-bit CRC (The least significant 8 bits of the status register cannot be written because they contain automatically generated CRC data)

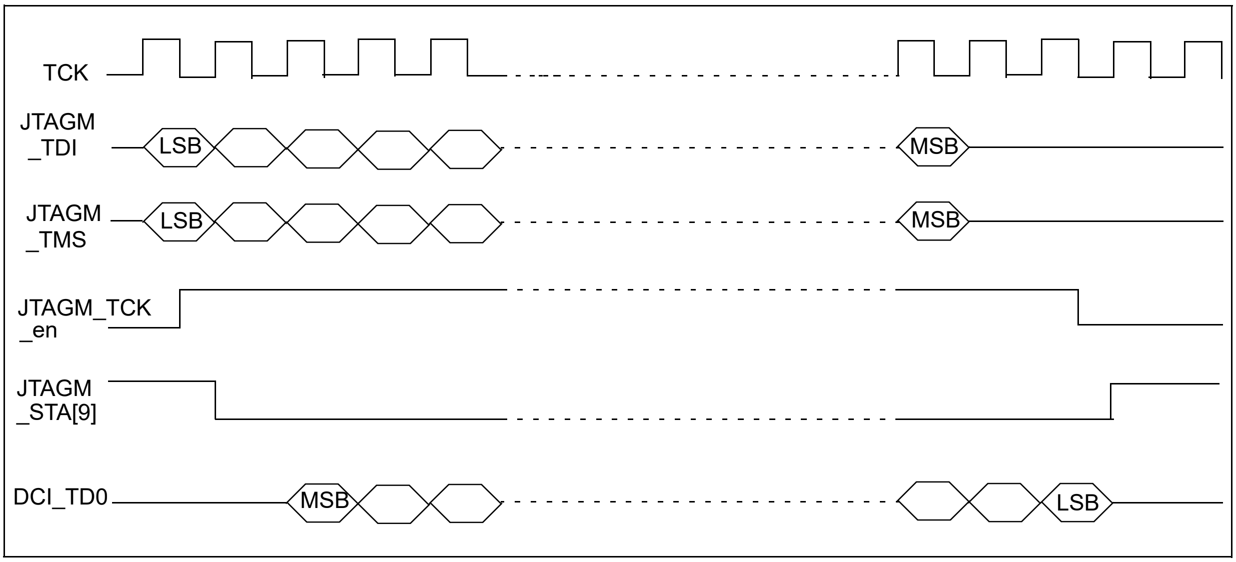
The masked data is transferred to the 32-bit configuration register and 32-bit status register. The JTAGM module responds to this configuration/status write by outputting a 96-bit payload to the parallel output port to channel number A. The 96-bit payload is made up of the 32-bit value of the configuration register plus 32 bits of zero-padding plus 32-bit value of the status register (including the least significant 8 bits which are an 8-bit CRC calculated on the previous 88 bits).

While the current messages are being transmitted from the second buffering (memory mapped registers), the first buffers are continuously filled up. The status register also includes status bits provided by the LFAST module and the DCI module. The JTAGM requests data from the LFAST only when at least one internal buffer is free.

63.2.8 JTAGM to DCI serial interface

Figure 1296 shows the data transfer timings between JTAGM and DCI.

Figure 1296. JTAGM to DCI serial interface diagram



63.2.9 JTAGM data handling and CRC calculation in LFAST mode

Figure 1297 shows the bit ordering when a data frame is transmitted from LFAST to JTAGM for subsequent transmission to DCI.

Figure 1297. Bit ordering for LFAST to JTAGM transfer

bit 127 to bit 68	bit 67 to bit 8	7	6	5	4	3	2	1	0
TMS (60 bits)	TDI (60 bits)	CRC (8 bits)							
bit 0 to bit 59	bit 0 to bit 59								
Last TMS bit transmitted by JTAGM ... First TMS bit transmitted by JTAGM	Last TDI bit transmitted by JTAGM ... First TDI bit transmitted by JTAGM								

Figure 1298 shows the bit ordering when a data frame is transmitted from JTAGM to LFAST consisting of TDO data received from DCI.

Figure 1298. Bit ordering for JTAGM to LFAST transfer

bit 95 to bit 36	bit 35 to bit 32	bit 31 to bit 8	7	6	5	4	3	2	1	0
TDO (60 bits)	0 pad (4 bits)	JTAGM_SR (24 bits)	CRC (8 bits)							
bit 0 to bit 59	bit 0 to bit 59	bit 31 to bit 8								
First TDO bit received by JTAGM ... Last TDO bit received by JTAGM										

The CRC calculation is described in the following example.

The data transferred from LFAST to JTAGM (in four frames) is as follows:

- D0—F7B3_D591
- D1—E6A2_C486
- D2—A2C4_80F7
- D3—B3D5_9183

These are supplied to CRC block starting from D0 to D3. The CRC calculation begins on D0 with the MSB (left bit) going first. D3 is supplied as B3D5_9100 (after masking CRC).

JTAGM extracts TMS and TDI information from the data received as follows:

- TMS = F7B3_D591_E6A2_C48
- TDI = 6A2C_480F_7B3D_591

JTAGM transmits the data to DCI starting from LSB (right bit going first). Corresponding to this TMS and TDI transmission, JTAGM receives TDO data from DCI and stores it as first bit to most significant location in TDO frame (shown in *Figure 1298*).

63.2.10 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

63.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

63.4 Memory map and register definition

Table 1297 contains the memory map of the JTAGM registers.

Table 1297. Memory map

Offset	Register	Location
0x00	Module Configuration Register (JTAGM_MCR)	Section 63.4.1.1
0x04	Status Register (JTAGM_SR)	Section 63.4.1.2
0x08	Data Out Register 0 (JTAGM_DOR0)	Section 63.4.1.3
0x0C	Data Out Register 1 (JTAGM_DOR1)	Section 63.4.1.4
0x10	Data Out Register 2 (JTAGM_DOR2)	Section 63.4.1.5
0x14	Data Out Register 3 (JTAGM_DOR3)	Section 63.4.1.6
0x18	Receive CRC Register (JTAGM_RXCRC)	Section 63.4.1.7
0x1C	Data Input Register 0 (JTAGM_DIR0)	Section 63.4.1.8
0x20	Data Input Register 1 (JTAGM_DIR1)	Section 63.4.1.9

63.4.1 Register descriptions

63.4.1.1 Module configuration register (JTAGM_MCR)

This register contains the status bits for the current transfer. Figure 1299 shows the format of the JTAGM_MCR.

Offset 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWRESET	evto0_sense		evto1_sense		evto_IE	evti0_assert	evti1_assert	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	inter_jtag_frame_timer						0	SIE	IIE	TCKSEL			jtagm_JCOMP	DTM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 1299. JTAGM_MCR register

The JTAGM_MCR register is described in Table 1298.

Table 1298. JTAGM_MCR register field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit. Writing a 0 has no effect. **Note:** After being written to “1” , this bit is auto cleared within 3 TCK cycles.
1:2 evto0_sense	Determines JTAGM response to activity on $\overline{\mathrm{EVTO}}0$ 00 No action is taken by the JTAGM in response to any $\overline{\mathrm{EVTO}}0$ edges (default setting) 01 evto0_edge status bit is set whenever a falling edge is detected on $\overline{\mathrm{EVTO}}0$ 10 evto0_edge status bit is set whenever a rising edge is detected on $\overline{\mathrm{EVTO}}0$ 11 evto0_edge status bit is set whenever a falling or rising edge is detected on $\overline{\mathrm{EVTO}}0$
3:4 evto1_sense	Determines JTAGM response to activity on $\overline{\mathrm{EVTO}}1$ line 00 No action is taken by the JTAGM in response to any $\overline{\mathrm{EVTO}}1$ edges (default setting) 01 evto1_edge status bit is set whenever a falling edge is detected on $\overline{\mathrm{EVTO}}1$ 10 evto1_edge status bit is set whenever a rising edge is detected on $\overline{\mathrm{EVTO}}1$ 11 evto1_edge status bit is set whenever a falling or rising edge is detected on $\overline{\mathrm{EVTO}}1$
5 evto_IE	Enables $\overline{\mathrm{EVTO}}$ triggered JTAGM interrupt 0 No JTAGM interrupt is generated In response to any $\overline{\mathrm{EVTO}}0$ or $\overline{\mathrm{EVTO}}1$ activity 1 JTAGM interrupt is generated when evto0_edge or evto1_edge are set
6 evti0_assert	EVTI0 Assert 0 Set jtagm_evti0 low/asserted 1 Set jtagm_evti0 high/de-asserted
7 evti1_assert	EVTI1 Assert 0 Set jtagm_evti1 low/asserted 1 Set jtagm_evti1 high/de-asserted **Note:** In the SW mode (when DTM =1) , regardless of the state of “evti1_assert” bit the internall “evti” signal remains de-asserted. Though this bit can be written, writing to this bit has no effect .
18:23 inter_jtag_frame_timer	TCK delay 000000 0 TCK cycle 000001 1 TCK cycle … 111111 63 TCK cycles
25 SIE	SPU Interrupt Enable 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer **Note:** Before enabling a JTAGM idle interrupt by setting JTAGM_MCR[IIE] = 0b1, set the Idle bit (JTAGM_SR[Idle]) to 0b1 to clear any previous interrupts.

Table 1298. JTAGM_MCR register field descriptions (continued)

Field	Description
27:29 TCKSEL	TCK Divider division factor control value. (TCK Divider divides the clock input to JTAGM IP to generate TCK out(input to DCI)). 000 TCK Divider division factor is 1 001 TCK Divider division factor is 2 010 TCK Divider division factor is 3 011 TCK Divider division factor is 4 100 TCK Divider division factor is 5 101 TCK Divider division factor is 6 110 TCK Divider division factor is 7 111 TCK Divider division factor is 8 **Note:** It should be taken care that the maximum frequency of TCK input to DCI, should be equal or less than half of system clock.
30 jtagm_JCOMP	JTAG reset 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data for JTAG transferred by LFAST 1 Data for JTAG transferred by software

63.4.1.2 Status register (JTAGM_SR)

63.4.1.2.1 LFAST mode

Figure 1300 shows the format of the JTAGM_SR in LFAST mode.

Offset 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	overrun	0	lfast_status						dci_status							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_edge	evto1_edge	SPU_INT_CLR	SPU_INT	CRC_err	Nexus_err	Idle	NR	CRC							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 1300. JTAGM_SR register in LFAST mode

63.4.1.2.2 SW mode

Figure 1301 shows the format of the JTAGM_SR in SW mode.

Offset 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	lfast_status						dci_status							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_edge	evto1_edge	SPU_INT_CLR	SPU_INT	CRC_err	Nexus_err	Idle	NR			0	0	0	0	0	0
W			SPU_INT_CLR						evto0_clr	evto1_clr						
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 1301. JTAGM_SR register in SW mode

The JTAGM_SR register is described in *Table 1299*.

Table 1299. JTAGM_SR register field descriptions

Field	Description
0 overrun	Set by hardware when the JTAGM overwrites the first Tx data with the second Tx data. Writing 1 clears this bit. This bit is set only in LFAST mode.
2:7 lfast_status	Status information from the LFAST to be used by software. The composition of these bits are as follows: bit 2 Indicates either data frame or ping frame was transmitted successfully bit 3 Indicates Tx data Interface not enabled and a frame is ready to be transmitted bit 4 Indicates Rx data FIFO overflow bit 5 Indicates reception of frame with invalid frame payload size bit 6 Indicates reception of frame with invalid ICLC code bit 7 Indicates reception of illegal LCT frame
8:15 dci_status	Status information from the DCI. The composition of these bits is as follows: bit 8 Reserved to 0 bit 9 DCI LVDS port enable bit 10 DCI JTAG port enable bit 11 DCI LVDS safe mode bit 12 DCI JTAG safe mode bit 13 DCI LVDS escape mode bit 14 DCI enable LFAST bit 15 DCI tool present , set when a debug tool is connected
16 evto0_edge	In LFAST mode (DTM = 0) a channel B message is sent on LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit 0 No activity sensed on $\overline{EVTO0}$ 1 Activity defined by evto0_sense detected on $\overline{EVTO0}$

Table 1299. JTAGM_SR register field descriptions (continued)

Field	Description
17 evto1_edge	In LFAST mode (DTM = 0) a channel B message is sent to the LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto1_clr bit 0 No activity sensed on $\overline{\text{EVTO1}}$ 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO1}}$
18 SPU_INT_CLR	Write 1 clears the SPU_INT This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit.
20 CRC_err	CRC error bit. Set when the calculated CRC does not match the last received CRC. This bit is cleared when the error packet is sent to the LFAST on channel B.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to ‘1’ when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read **Note:** In LFAST mode(when DTM =0) , this bit has no effect and acts as a “dont care” . This bit should not be used by the application in the LFAST mode
24:31 CRC	Calculated CRC. CRC calculated on the information received. LFAST Mode: Bit 24:31 reads the CRC. Write has no effect. SW Mode: – Bit 24:31 reads as 0x00. Writing a 0 to any bit have no effect. – Writing 1 on bit 24 clears evto0_edge – Writing 1 on bit 25 clears evto1_edge

63.4.1.3 Data out register 0 (JTAGM_DOR0)

Figure 1302 shows the format of the JTAGM_DOR0.

Offset 0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TMS_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1302. JTAGM_DOR0 register

The JTAGM_DOR0 register is described in *Table 1300*.

Table 1300. JTAGM_DOR0 register field descriptions

Field	Description
0:31 TMS_HIGH	Higher word of data for TMS, bits 59-28. **Note:** If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.

63.4.1.4 Data out register 1 (JTAGM_DOR1)

Figure 1303 shows the format of the JTAGM_DOR1.

Offset 0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TMS_LOW																												0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1303. JTAGM_DOR1 register

The JTAGM_DOR1 register is described in *Table 1301*.

Table 1301. JTAGM_DOR1 register field descriptions

Field	Description
0:27 TMS_LOW	Lower word of data for TMS, bits 27–0. TMS_LOW[0] is shifted out first. **Note:** If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.

63.4.1.5 Data out register 2 (JTAGM_DOR2)

Figure 1304 shows the format of the JTAGM_DOR2.

Offset 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDI_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1304. JTAGM_DOR2 register

The JTAGM_DOR2 register is described in *Table 1302*.

Table 1302. JTAGM_DOR2 register field descriptions

Field	Description
0:31 TDI_HIGH	Higher word of data for TDI, bits 59–28. **Note:** If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.

63.4.1.6 Data out register 3 (JTAGM_DOR3)

Figure 1305 shows the format of the JTAGM_DOR3.

Offset 0x14 Access: User read/write

	0–27	28	29	30	31
R	TDI_LOW	0	0	0	Send
W					
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0	0	0	0

Figure 1305. JTAGM_DOR3 register

The JTAGM_DOR3 register is described in *Table 1303*.

Table 1303. JTAGM_DOR3 register field descriptions

Field	Description
0:27 TDI_LOW	Lower word of data for TDI, bits 27–0. TDI_LOW[0] is shifted out first. **Note:** If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

63.4.1.7 Receive CRC register (JTAGM_RXCRC)

Figure 1306 shows the format of the JTAGM_RXCRC register.

Offset 0x18 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24–31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CRC
W																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0

Figure 1306. JTAGM_RXCRC register

The JTAGM_RXCRC register is described in *Table 1304*.

Table 1304. JTAGM_RXCRC register field descriptions

Field	Description
24:31 CRC	Received CRC. Refers to the CRC of the received LFAST message.

63.4.1.8 Data input register 0 (JTAGM_DIR0)

Figure 1307 shows the format of the JTAGM_DIR0.

Offset 0x1C Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TDO_LOW																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1307. JTAGM_DIR0 register

The JTAGM_DIR0 register is described in *Table 1305*.

Table 1305. JTAGM_DIR0 register field descriptions

Field	Description
0:31 TDO_LOW	Lower word of data received on TDO, bits 0–31

63.4.1.9 Data input register 1 (JTAGM_DIR1)

Figure 1308 shows the format of the JTAGM_DIR1.

Offset 0x20 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TDO_HIGH																												0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1308. JTAGM_DIR1 register

The JTAGM_DIR1 register is described in *Table 1306*.

Table 1306. JTAGM_DIR1 register field descriptions

Field	Description
0:27 TDO_HIGH	Higher word of data received on TDO, bits 32–59

64 Debug Zipwire

64.1 Overview

The Debug SIPI and Debug LFAST modules work together as a single unit called Debug Zipwire. The Debug LFAST portion of the two modules allows for high speed debugging capabilities.

The Debug LFAST operates in slave mode configuration. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

64.2 Debug Zipwire Overview

Dedicated Zipwire functionality is available for debug use allowing support for increased bandwidth read/write operations to the chip memory space performed over the debug interface. This Debug Zipwire function is available on both production and ED devices. The modules used to support Debug Zipwire are the Debug SIPI module, the Debug LFAST module and the LFAST Switch.

The Debug Zipwire interface allows a connected tool to perform 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address in the microcontroller. The Zipwire architecture is fully pipelined and supports multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at 320 Mbaud using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate 20 MHz reference clock for a total of five pins.The Debug Zipwire interface does not support streaming mode commands.

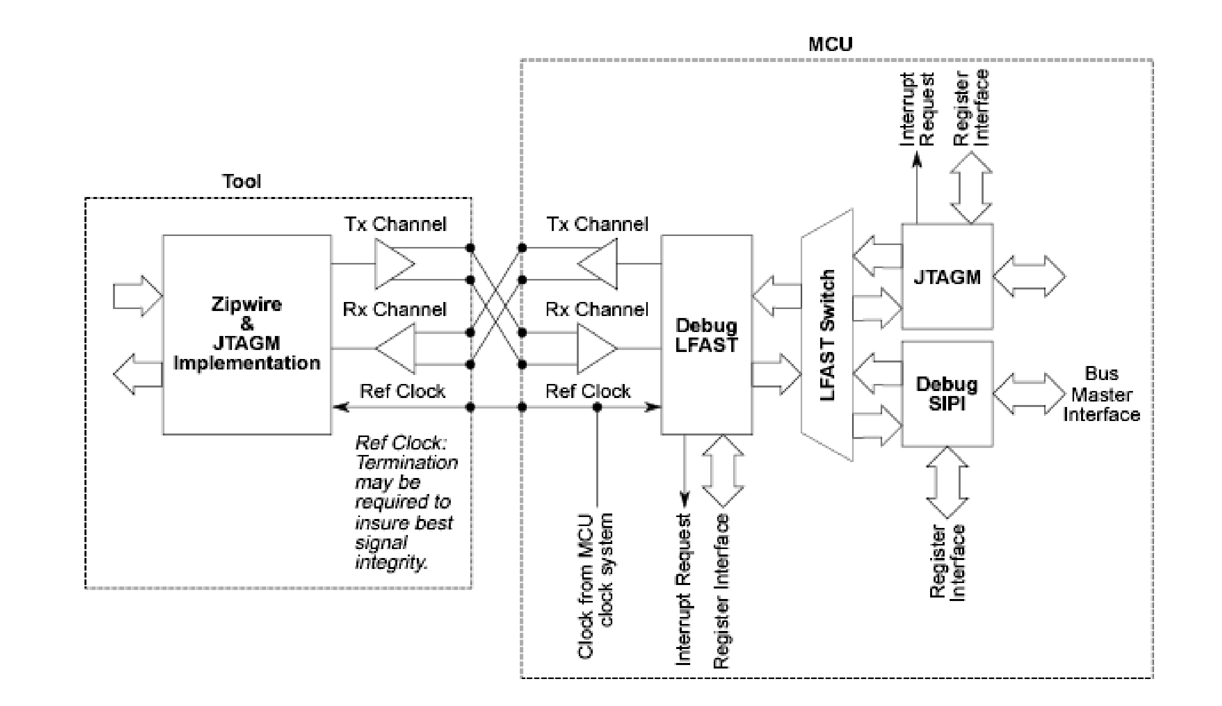
For single random access, 32-bit read, from any 32-bit address location, the latency is approximately 1 µs.

64.3 Debug Zipwire Block Diagram

Figure 1309 shows a tool connected to the MCU via five wires. These five pins are shared with the standard JTAG pins, but for simplicity this is not shown in the diagram. The MCU Debug Zipwire only supports the Target mode of Zipwire. The Debug SIPI Bus Master Interface is used for all Target Mode transactions. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST Reference clock coming from the MCU clock system.

Figure 1309. Debug Zipwire connection diagram



64.4 Architecture

Debug Zipwire is composed of several modules and system resources. The physical layer is LVDS with 1.2 V common mode voltage and 200 mV swing.The pads are shared with JTAG signals.

The transport layer is a protocol called LFAST. LFAST is an asynchronous prototcol, using non-return to zero encoding. The LFAST protocol is composed of the following:

- A fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the incoming data.
- Followed by an 8-bit LFAST header, that defines the channel number and the size of the LFAST payload.
- Finally the payload, which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI). Debug SIPI runs on top of Debug LFAST and is fully encapsulated within the Debug LFAST payload. Debug SIPI uses three fixed sizes of Debug LFAST payload in Debug Zipwire:

- 32-bit
- 64-bit
- 96-bit

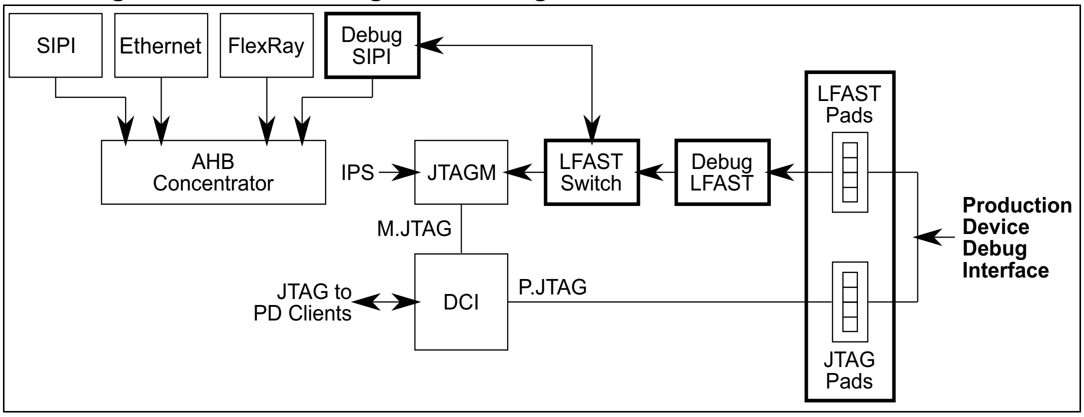
The SIPI protocol implements a suite of commands initiated by a tool, for reading and writing any 32-bit address location in a connected MCU. Only the tool can initiate commands. The MCU only responds to commands initiated by the tool and never creates any unsolicited messages.

For Debug Zipwire, a reduced version of the SIPI module called Debug SIPI is used. The Debug SIPI module implements only the recipient part of the SIPI protocol (called Target mode). The Debug SIPI module is a bus master on the low speed XBAR. As the target MCU for a command from the inititating tool, Debug SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

64.5 Debug Zipwire implementation

Figure 1310 shows the modules on the production device which support the Debug Zipwire function, along with their interconnections to the debug interface signals.

Figure 1310. Block diagram showing connections for Production Device



Internal connections to other debug modules which are used to support other debug interface protocols such as JTAG and JTAG over LFAST are also shown. The Debug SIPI module is connected to the Debug LFAST module via an LFAST switch module, which controls routing of debug LFAST traffic to either the Debug SIPI or the JTAGM module.

64.6 LFAST clocking

The LFAST implementation includes a dedicated PLL for multiplying a low speed clock source up to 320 MHz for the transmission and reception of data.

This PLL is used to support Debug Zipwire so care must be taken to ensure that any tool configuration of this resource is compatible with application usage of Interprocessor Bus Zipwire.

The high speed asynchronous nature of LFAST, requires that both the local node and remote node, operate from the same, stable, low jitter clock sources. This common low speed clock is referred to as the Reference Clock.

The MCU includes clock multiplexing to allow the source of the LFAST reference clock to come from one of three sources via a programmable clock divider:

- XOSC
- Output of PLL0 (PLL0:PHI)
- External pin

The most expedient source for the reference clock is the 20 MHz crystal oscillator (XOSC) from one MCU. This 20 MHz clock can be used as the LFAST Reference clock for the local

LFAST. This same 20 MHz clock must be exported from the local MCU to the remote MCU, to be used as the LFAST reference at the remote MCU.

The 20 MHz clock may be used in the following ways:

- Shared between the MCUs using a dedicated reference clock pin on each MCU
- Divided by 2 and shared as a 10 MHz reference
- Shared as the fundamental clock for the MCU system clock and the LFAST reference clock

The LFAST PLL for generating the 320 MHz clock, is shared between LFAST for SIPI.

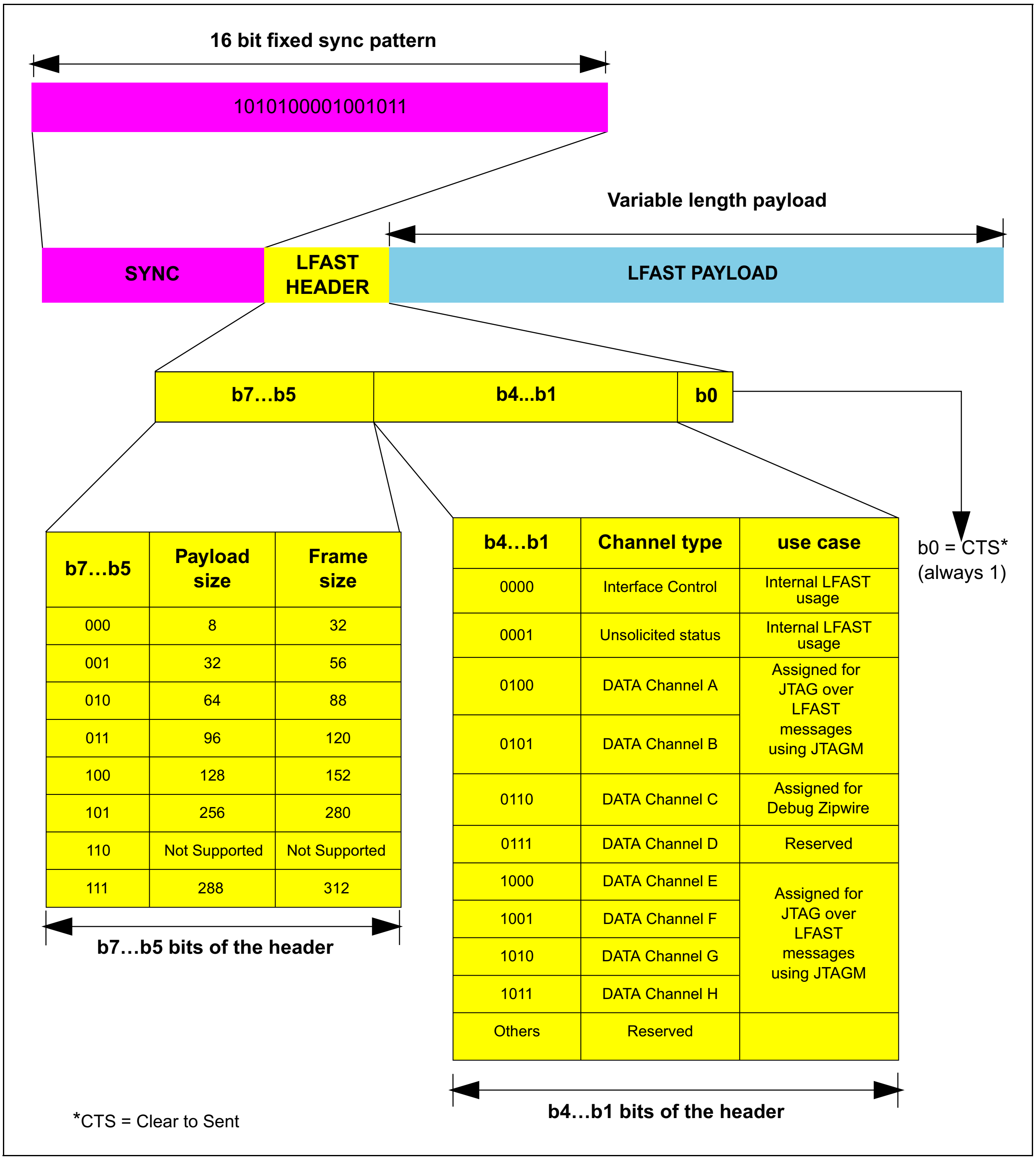
64.7 LFAST switch

Incoming messages on the Debug LFAST interface can be either for JTAG over LFAST use through the JTAGM or debug Zipwire messages. Differentiation of these messages is via use of LFAST channel number. DATA channel C is used to identify high speed Debug SIPI accesses, while JTAG over LFAST messages are supported using LFAST channels A, B, E, F, G, H as shown in *Figure 1311*

The LFAST SWITCH module is used to direct LFAST messages based on the above scheme, with JTAG over LFAST messages being routed to the JTAGM IP block and debug Zipwire messages being routed to the Debug SIPI block.

The LFAST switch has no accessible registers and requires no user configuration.

Figure 1311. LFAST channel allocation



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	Not Supported	Not Supported
111	288	312

b4…b1	Channel type	use case
0000	Interface Control	Internal LFAST usage
0001	Unsolicited status	Internal LFAST usage
0100	DATA Channel A	Assigned for JTAG over LFAST messages using JTAGM
0101	DATA Channel B	
0110	DATA Channel C	Assigned for Debug Zipwire
0111	DATA Channel D	Reserved
1000	DATA Channel E	Assigned for JTAG over LFAST messages using JTAGM
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

64.8 Debug SIPI module

The debug SIPI module is used to support high bandwidth read/write accesses over the LFAST debug interface. It is dedicated for debug use and is entirely separate from the SIPI module used to support interprocessor bus zipwire interface.

The debug SIPI characteristics are shown in the table below:

Table 1307. Debug SIPI feature set

Feature	Debug implementation
Number of channels supported	1
Initiator support	No
Target support	Yes
Block transfer support	No
Chip XBAR connection	M1 on XBAR_1

The debug SIPI module issues read and write accesses via the AHB concentrator connected to physical master port 2 on XBAR_1, which is shared with the Ethernet and Zipwire SIPI bus masters.

The debug SIPI memory mapped registers are accessible via PBRIDGE_0 (off platform - slot 10) with base address FFFD_4000h.

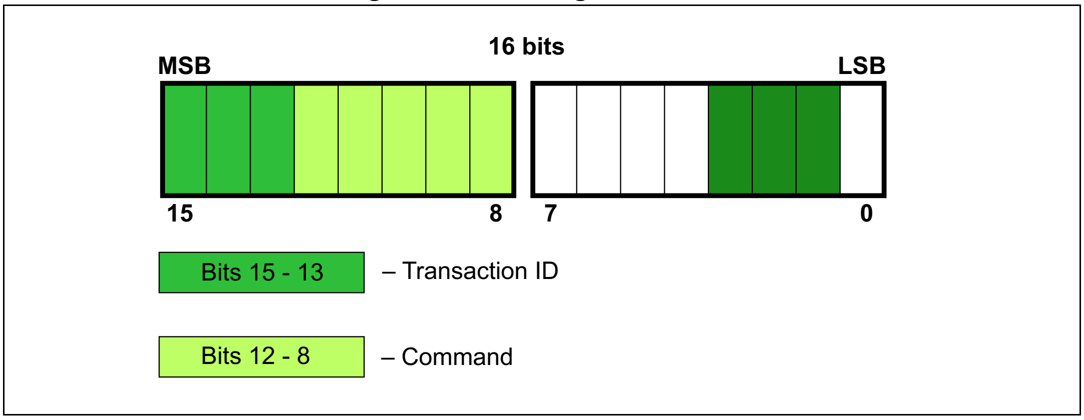
The debug SIPI has Logical Bus Master Assignment of 6 for SMPU0 and SMPU1, an NXMC trace source ID (SRC) of 1101b and a trace master ID (MASTER) = 1011b.

The debug SIPI supports only target mode SIPI operations on a single channel with a reduced command set. Commands supported include read (8/16/32 bits) and write (8/16/32 bits) along with the appropriate responses. Other SIPI commands are unimplemented as noted in *Table 1308*.

The debug SIPI will be capable of:

- receiving a 64-bit command on LFAST channel C and responding with either a 64-bit response or 32-bit error response on LFAST channel C.
- receiving a 96-bit command on LFAST channel C and responding with either a 32-bit acknowledge or 32-bit error response on LFAST channel C.

Figure 1312. Debug SIPI header



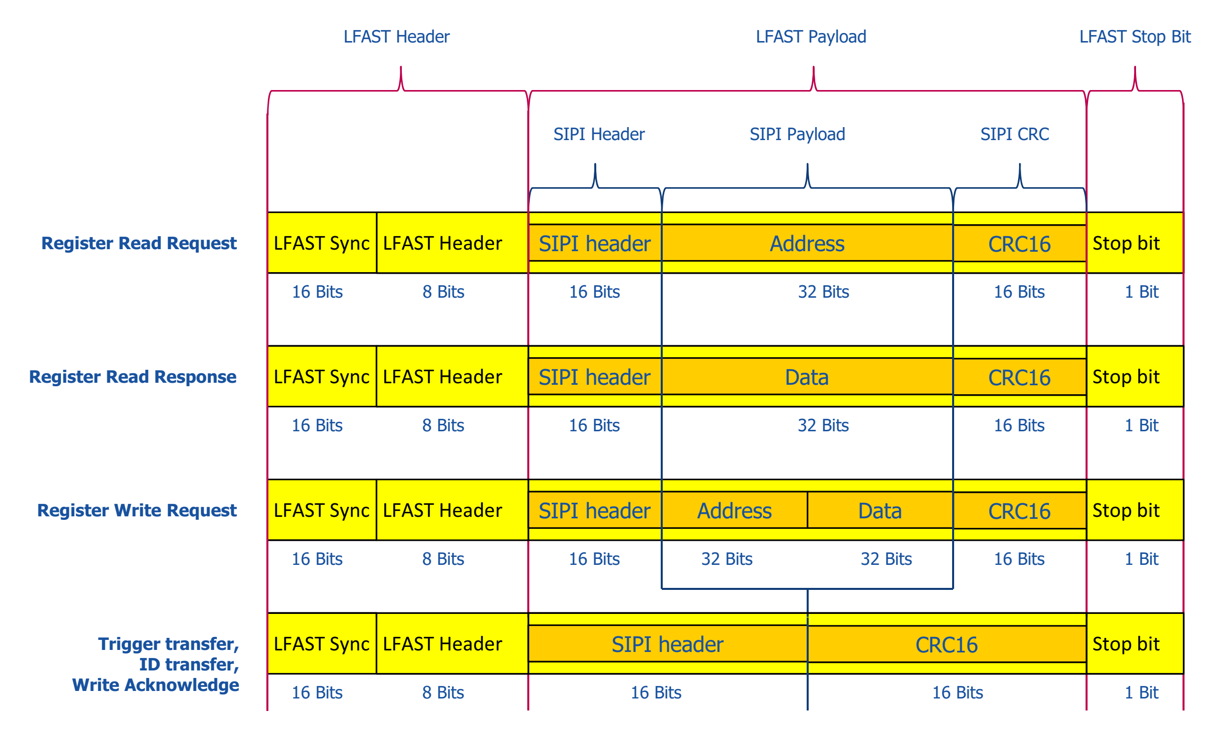
Note: See Figure 1311 for the description of bits[7:0]

Table 1308. SIPI header command coding

Bits[12:8] (hex)	Command	Payload Size	Message Direction at chip (SIPI Target)
00	Read 8 bits (Received by SIPI from LFAST)	64	Rx
01	Read 16 bits (Received by SIPI from LFAST)	64	Rx
02	Read 32 Bits (Received by SIPI from LFAST)	64	Rx
03	Reserved	—	—
04	Write 8 bits with ACK (Received by SIPI from LFAST)	96	Rx
05	Write 16 bits with ACK (Received by SIPI from LFAST)	96	Rx
06	Write 32 bits with ACK (Received by SIPI from LFAST)	96	Rx
07	Reserved	—	—
08	ACK – OK (Issued to LFAST from SIPI)	32	Tx
09	ACK – Fault (Issued to LFAST from SIPI)(1)	32	Tx
0A	Read Answer – OK (Issued to LFAST from SIPI)	64	Tx
0B	Reserved	—	—
0C	Unimplemented – Trigger comm and with ACK	32	—
0D	Reserved	—	—
…	…	…	…
11	Reserved	—	—
12	ID Register Read Request	32	Rx
13	Reserved	—	—
…	…	…	…
16	Reserved	—	—
17	Unimplemented – Stream Data with ACK (32 bytes)	288	—
18	Reserved	—	—
…	…	…	…
1F	Reserved	—	—

1. ACK – Fault is also referred to as “NACK” and “ACK Error” in the HSSL/SIPI Interprocessor Bus Protocol Specification.

Figure 1313. Zipwire message format



Note: Identification Register Read is the same as the Register Read response format.

64.9 Debug Zipwire interconnections

Debug LFAST has the following connections:

- Tx and Rx connections to the pads via the SIUL and the MSCR registers.
- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers.
- Tx Data Port/Rx Data Port — Connected to the LFAST Switch. This allows LFAST data to be dynamically routed to/from either Debug SIPI for Debug Zipwire operation or JTAGM for JTAG over LFAST operation.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

LFAST Switch has the following connections:

- Tx Data Port/Rx Data port directly connected to Debug SIPI.
- Tx Data Port/Rx Data port directly connected to JTAGM.
- Tx Data Port/Rx Data port directly connected to Debug LFAST.
- LFAST switch has no IPS interface; no registers; no interrupts.

Debug SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers and SIPI Interface Registers. For the Debug SIPI, a restricted set of registers is implemented. In use, no configuration of these registers is needed from either s/w or a connected tool, as the default settings should be usable for typical operation.

- Crossbar master port — Allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port — Directly connected to the LFAST module. Allows received data to be efficiently transferred from LFAST and transmit data to be transferred to LFAST.

64.10 Debug Zipwire performance

Two aspects of performance are considered:

- Bandwidth — The rate at which data can be read or written between two nodes. It assumes that read or write commands from the Inititator node, can be sent continuously at the highest speed the Target node can consume those commands.
- Latency — The time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

Bandwidth is the rate at which data can be read or written between the tool and the MCU. It assumes that read or write commands from the Inititator node (the Tool), can be sent continuously at the highest speed the Target node can consume those commands.

Latency is the time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

64.10.1 Read performance

A Zipwire read operation consists of three stages:

- Inititator sends SIPI Read command to Target.
- Target parses the received command and runs a master bus cycle to read the data.
- Target sends the SIPI Read response back to the Initiator.

A SIPI Read command consists of:

- 16-bit header
- 32-bit read address
- 16-bit CRC
- 64 bits total

A SIPI Read response consists of:

- 16-bit header
- 32-bit read data
- 16-bit CRC
- 64 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 64-bit payload
- 1-bit stop bit
- 89 bits total

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 ns.

Therefore, an 89-bit LFAST transmission of a SIPI 64-bit Read command or 64-bit Read response takes 278 ns.

The SIPI module takes thirteen system clock cycles to parse a command and create a Read response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being read.

Assumptions:

- Average of 6 × slow XBAR cycles to read different memories.
- Slow XBAR clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, time for SIPI to read an address location is:

(13 × 50 MHz clocks) + (6 × 100 MHz clocks) = 320 ns

64.10.1.1 Read bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Read response.

The time to transmit a command or response is less than the time to process the command. Therefore, the bandwidth is limited by the time to process the message.

Read Bandwidth = 4 bytes in 320 ns = 12.5 MB/second

64.10.1.2 Read latency

The latency is the time taken to send a Read command, process the command, and send a Read response.

Read Latency = 278 ns + 320 ns + 278 ns = 876 ns

64.10.2 Write performance

A Zipwire write operation consists of three stages:

- Inititator sends SIPI Write command to Target.
- Target parses the received command and runs a master bus cycle to write the data.
- Target sends the SIPI Write response back to the Initiator.

A SIPI Write command consists of:

- 16-bit header
- 32-bit write address
- 32-bit write data
- 16-bit CRC
- 96 bits total

A SIPI Write acknowledge consists of:

- 16-bit header
- 16-bit CRC
- 32 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 96-bit payload (command) or 32-bit (acknowledge)
- 1-bit stop bit
- 121 bits total (command) or 57 bits (acknowledge)

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 ns

Therefore, the time for an LFAST transmission is:

- 121-bit LFAST transmission of a SIPI 96-bit Write Command takes 378 ns
- 57-bit LFAST transmission of a SIPI 32-bit Write Acknowledge takes 178 ns

The SIPI module takes thirteen system clock cycles to parse a command and create a Write response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being written.

Assumptions:

- Average of 5 × slow XBAR cycles to write different memories.
- Slow XBAR clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, the time for SIPI to write an address location is:

(13 × 50 MHz clocks) + (5 × 100 MHz clocks) = 310 ns

64.10.2.1 Write bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Write Acknowledge response.

The time to transmit the command is longer than the time to process the command, or the time to transmit the Write Acknowledge. Therefore, the bandwidth is limited by the time to transmit the command.

Write Bandwidth = 4 bytes in 378 ns = 10.6 MB/second

64.10.2.2 Write latency

The latency is the time taken to send a Write command, process the command, and send a Write Acknowledge.

Write Latency = 378 ns + 310 ns + 178 ns = 866 ns

65 Debug Serial Interprocessor Interface (SIPI)

65.1 Introduction

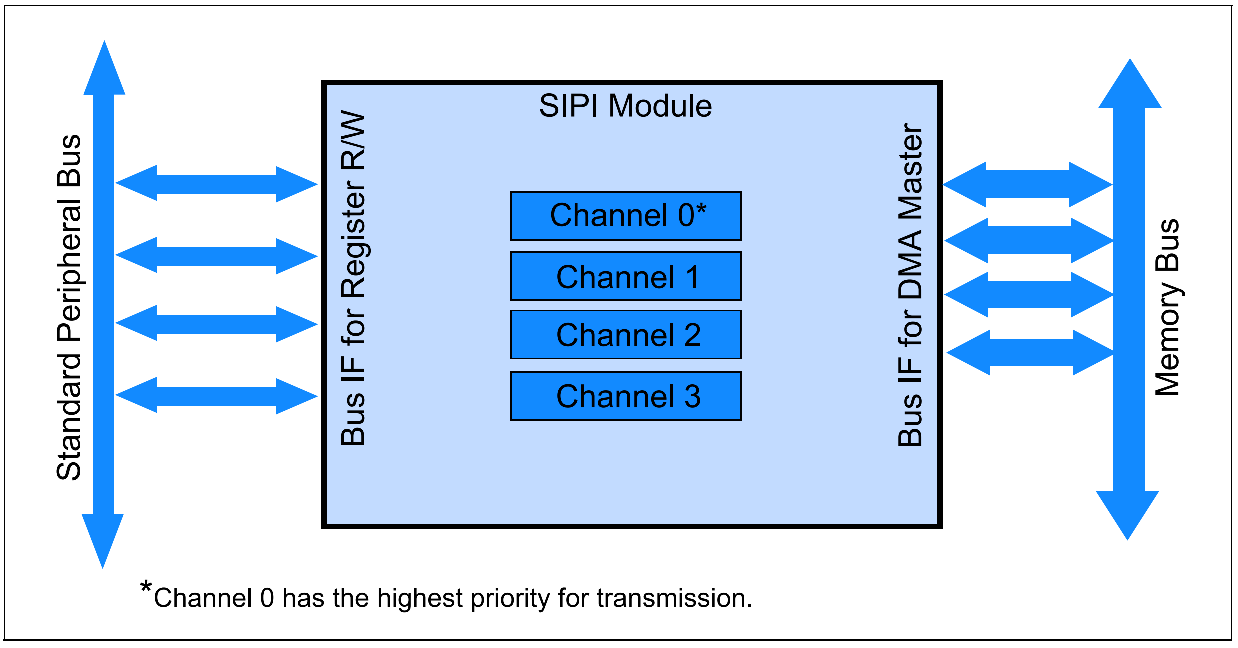
The Debug Serial Interprocessor Interface (SIPI) is an application layer protocol that runs on top of the LFAST module. It is used by the local device to access the shared memory of a remote device. Debug SIPI defines point-to-point full duplex communication between two nodes, where LFAST works as a physical medium of communication between both the devices.

Debug SIPI implements the Target mode of a communication protocol in hardware. Debug SIPI can interpret 'read' and 'write' commands received from an LFAST module and convert them into bus accesses to local memory; then send back to the LFAST module either read data or a write acknowledge response. This provides a mechanism for an external device, such as an external tool, to read and write the address map of the local MCU. The bandwidth of these read/write commands is approximately 10 MBytes/second when accessing random addresses using 32-bit transfers.

65.2 Overview

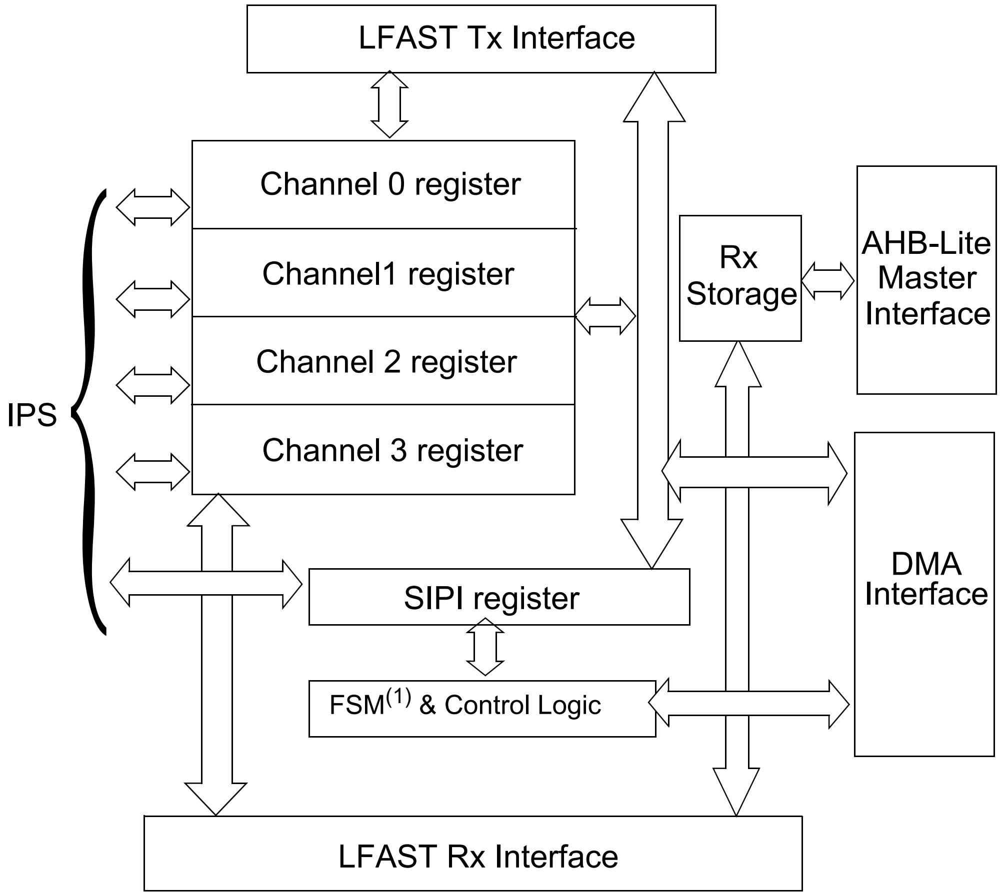
Debug SIPI can access memory mapped resources directly through its AHB master interface. Debug SIPI has four channels, 0, 1, 2, 3. Payload width for all channels is 32 bits.

Figure 1314. SIPI module



65.3 Debug SIPI block diagram

Figure 1315. SIPI block diagram



1. Finite State Machine

65.4 Feature description

This section describes the features of the Debug SIPI module.

65.4.1 Main features

- Supports 8, 16 and 32-bit read and write commands
- Command/response includes a 16-bit CRC to ensure data integrity
- Command/response includes a 2-bit tag to allow tracking of data to commands in a pipelined system
- Command/response includes a 3-bit channel identifier to allow multiple Initiators to send commands to a single target node
- Command/response includes a 16-bit header that defines, the function, channel and tag
- Every command generates a response to allow a 'sliding window' flow control protocol to be implemented
- AHB master interface that is used by target node to access memory mapped resources

65.4.2 Standard features

- IPS bus interface
- AHB master interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

65.5 Debug SIPI operation from reset

When the Debug SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

65.6 Functional description

65.6.1 External signals

The Debug SIPI has no chip external signals.

65.6.2 Frame format

All frames have the same general format:

- 16-bit header
- Address, address and data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID.

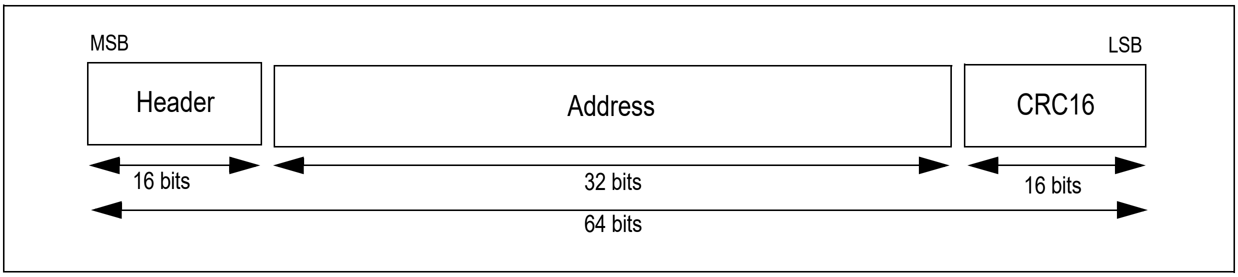
The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

Note: SIPI module architecture is always big endian but the storage in host memory depends on endianness architecture of host memory.

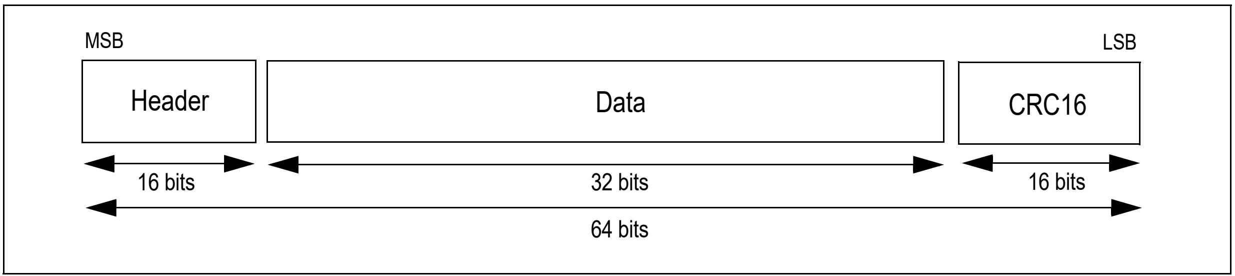
65.6.2.1 Register read request

Figure 1316. SIPI register read request



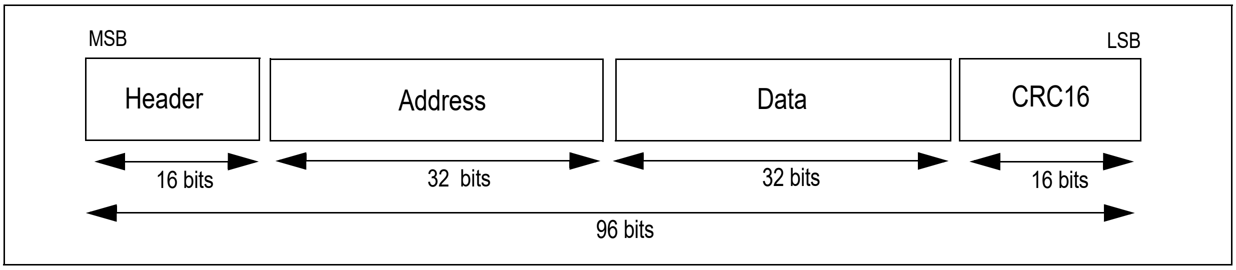
65.6.2.2 Register read response, ID request response

Figure 1317. SIPI read response



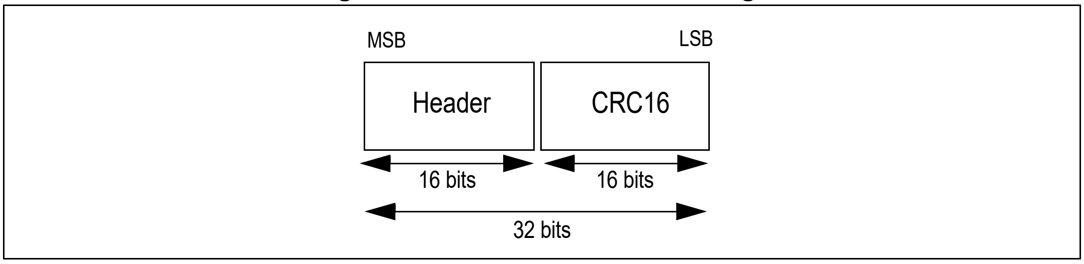
65.6.2.3 Register write request

Figure 1318. SIPI register write request



65.6.2.4 ID transfer, write acknowledge

Figure 1319. SIPI write acknowledge



65.6.2.5 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

65.6.2.5.1 Debug SIPI header coding

Figure 1320, Table 1309, and *Figure 1321* show how the Debug SIPI header bits are coded.

Figure 1320. SIPI header coding

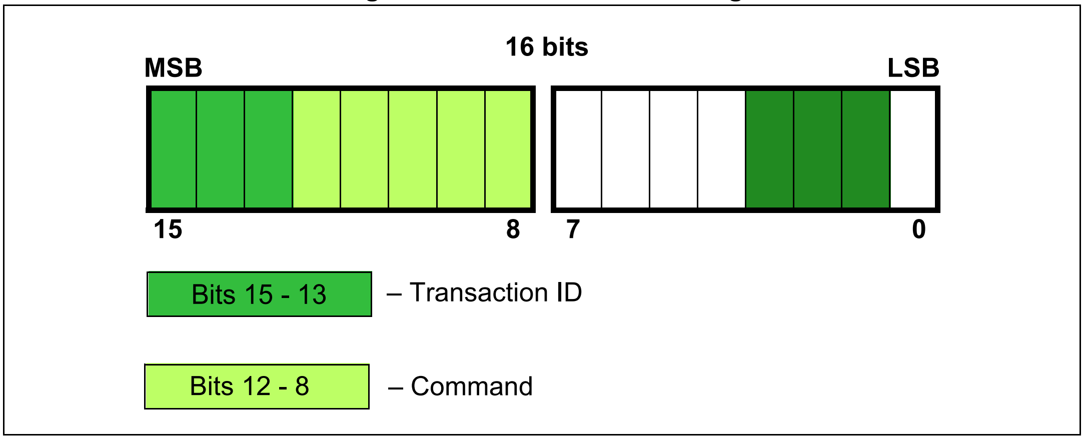


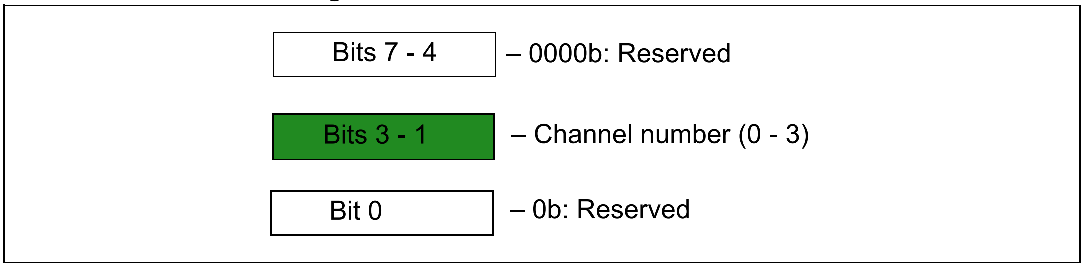
Table 1309 shows the command coding for the bits[12:8] of the header.

Table 1309. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
0x00	Read 8 bits	64
0x01	Read 16 bits	64
0x02	Read 32 Bits	64
0x03	Reserved	—
0x04	Write 8 bits with ACK	96
0x05	Write 16 bits with ACK	96
0x06	Write 32 bits with ACK	96
0x07	Reserved	—
0x08	ACK – OK	32
0x09	ACK – Fault	32
0x0A	Read Answer – OK	64
0x0B	Reserved	—
0x0C	Reserved	—
0x0D	Reserved	—
…	…	…
0x11	Reserved	—
0x12	ID Register Read Request	32
0x13	Reserved	—
…	…	…
0x16	Reserved	—
0x17	Reserved	—
0x18	Reserved	—
…	…	...
0x1F	Reserved	—

Figure 1321 shows the content of bits[7:0] in the SIPI header.

Figure 1321. SIPI header channel field



65.6.2.6 Address field

The address field is 32 bits wide with the MSB transmitted first.

65.6.2.7 Payload field

Table 1310 shows the payload sizes of LFAST frames.

Table 1310. Payload size of LFAST channel frame

LFAST code for payload size	Debug SIPI Code	LFAST payload size (bits)(1)	LFAST payload size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16
101b	101b	256	32
111b	111b	288	36

1. Only 32, 64, 96 and 288 bits sizes are used for SIPI.

Table 1311 shows the coding of LFAST channel for a given Debug SIPI channel code.

Table 1311. Converted coding of LFAST channel code for Debug SIPI headers

LFAST channel code	Debug SIPI channel code	Data channel(1)
0100b	100b	Data channel A
0101b	101b	Data channel B
0110b	110b	Data channel C
0111b	111b	Data channel D
1000b	000b	Data channel E (not used by Debug SIPI)
1001b	001b	Data channel F (not used by Debug SIPI)
1010b	010b	Data channel G (not used by Debug SIPI)
1011b	011b	Data channel H (not used by Debug SIPI)

1. Debug SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
Debug SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
Debug SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
Debug SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

65.6.2.8 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). The data stream is transmitted with the MSB sent first.

65.6.2.8.1 CRC examples

Example 1 – 32-bit write

- Header = 0x260A
- Address = 0x1122_3344
- Data = 0xCCDD_EEFF
- CRC = 0xBF7D

Example 2 – 32-bit read

- Header = 0x420C
- Address = 0x89AB_CDEF
- CRC = 0x6B80

Example 3 – Event command

- Header = 0x6C0E
- CRC = 0xB286

65.7 Transfer types

This section describes the available transfer types of the Debug SIPI module. The Debug SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

65.7.1 Read transfer

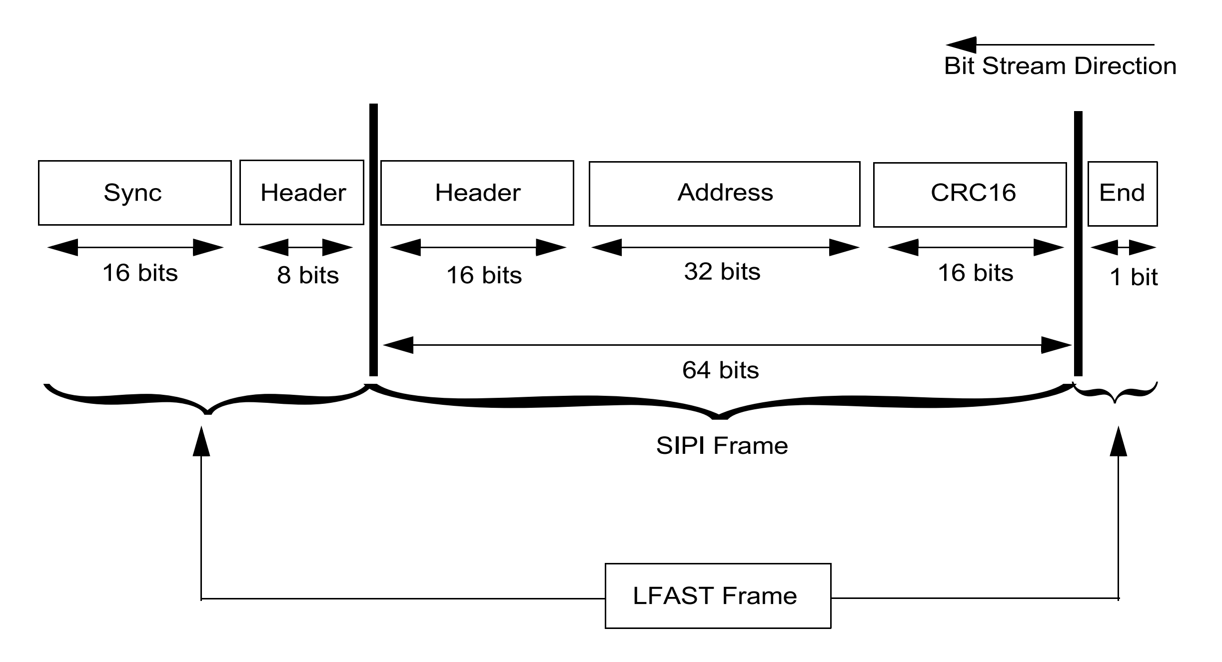
A Read transfer can be of two types:

- Read request transfer (at initiator node)
- Read response transfer (at target node)

65.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in *Figure 1322*.

Figure 1322. Read request transfer (showing LFAST frame encapsulation)



65.7.2 Register read answer transfer

In response to a read request, Debug SIPI sends header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (refer to *Figure 1323*). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (refer to *Figure 1324*).

Figure 1323. Read answer transfer (LFAST frame encapsulation is not shown)

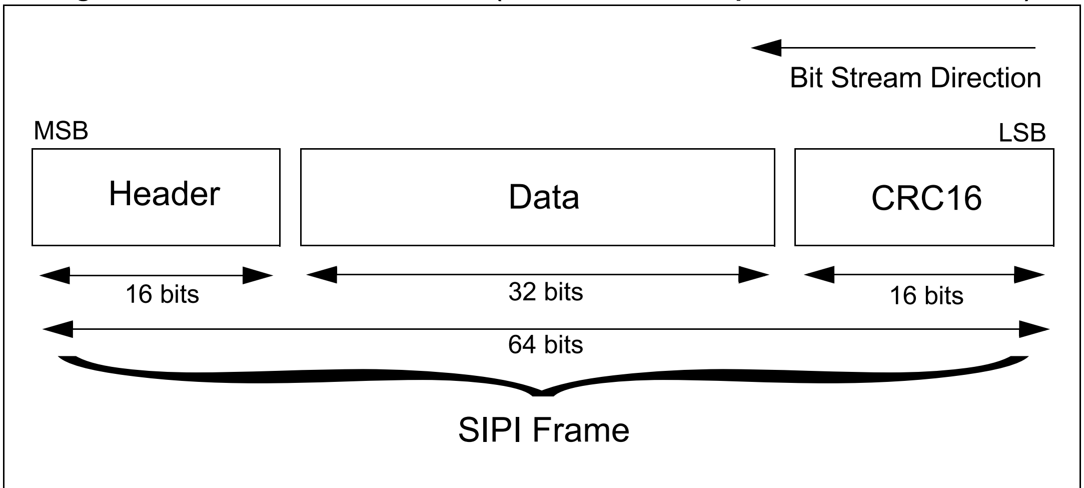
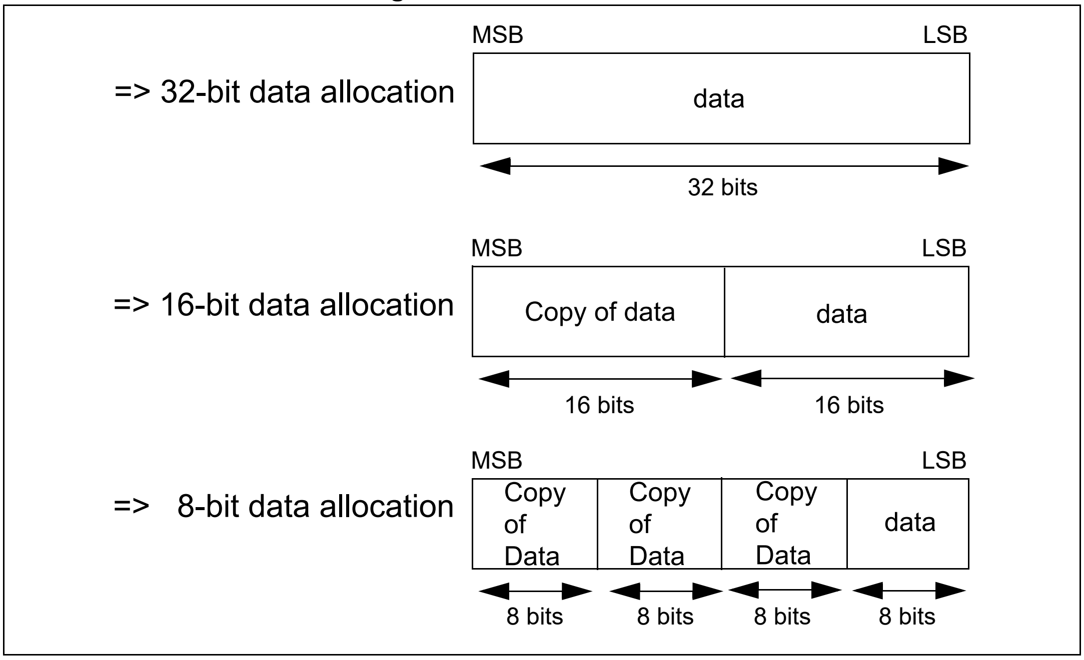


Figure 1324. Data allocation



Note: For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables (big endian):

Bits[1:0]:

- *00: Byte 3 enabled (MSB)*
- *01: Byte 2 enabled*
- *10: Byte 1 enabled*
- *11: Byte 0 enabled (LSB)*

For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable (big endian).

Bit[0]:

- *0: Half-word 1 enabled (MSB)*
- *1: Half-word 0 enabled (LSB)*

Note: Byte Ordering depends on the address written in CAR and transfer width:
Example:

- *CDR content: 0x12345678*
- *CAR content: see table columns*

Table 1312. Target Memory content dependent on CAR content

	Target Memory Byte Address	CAR = 0x02040	CAR = 0x02041	CAR = 0x02042	CAR = 0x02043
32-bit transfer	0x02040	12	12	12	12
	0x02041	34	34	34	34
	0x02042	56	56	56	56
	0x02043	78	78	78	78

Table 1312. Target Memory content dependent on CAR content (continued)

	Target Memory Byte Address	CAR = 0x02040	CAR = 0x02041	CAR = 0x02042	CAR = 0x02043
16-bit transfer	0x02040	12	56	xx	xx
	0x02041	34	78	xx	xx
	0x02042	xx	xx	12	56
	0x02043	xx	xx	34	78
8-bit transfer	0x02040	12	xx	xx	xx
	0x02041	xx	34	xx	xx
	0x02042	xx	xx	56	xx
	0x02043	xx	xx	xx	78

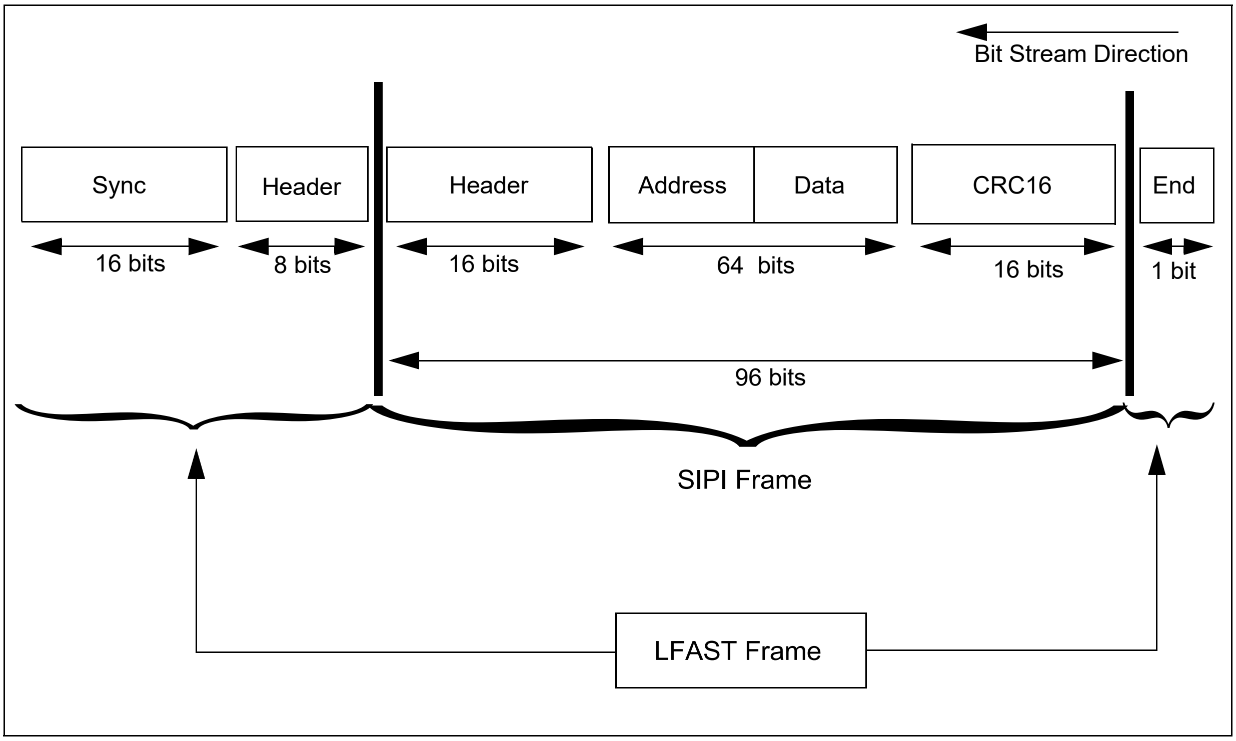
65.7.3 Register write transfer

Register write transfer can be of type:

- Normal write transfer – channels 0, 1, 2 and 3

A Register write transfer can be done through Normal write transfer - channels 0, 1, 2 and 3 (refer to *Section 65.7.3.1: Normal write transfer*) as shown in *Figure 1325*.

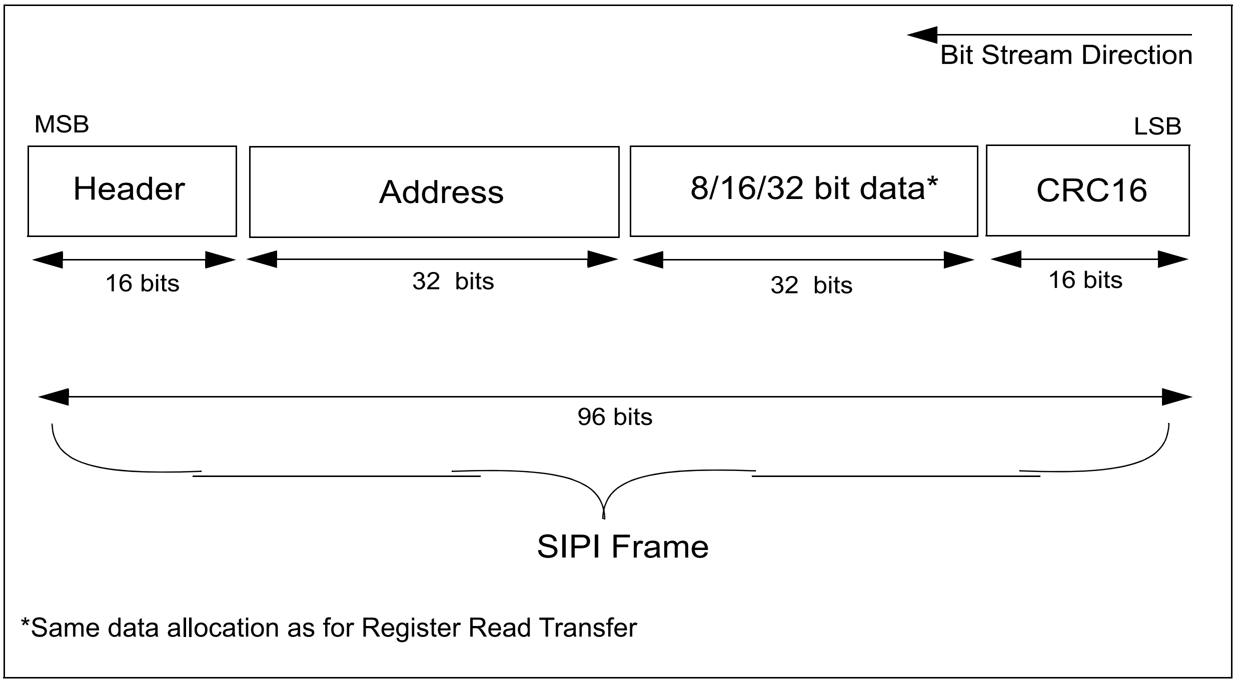
Figure 1325. Register write transfer (showing LFAST frame encapsulation)



65.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and 16-bit CRC as shown in *Figure 1326*.

Figure 1326. Write transfer (LFAST frame encapsulation is not shown)



Note: In the case of an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as a byte enable:

Bits[1:0]:

- *00 – byte 3 enabled (MSB)*
- *01 – byte 2 enabled*
- *10 – byte 1 enabled*
- *11 – byte 0 enabled (LSB)*

In the case of a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as a half-word enable:

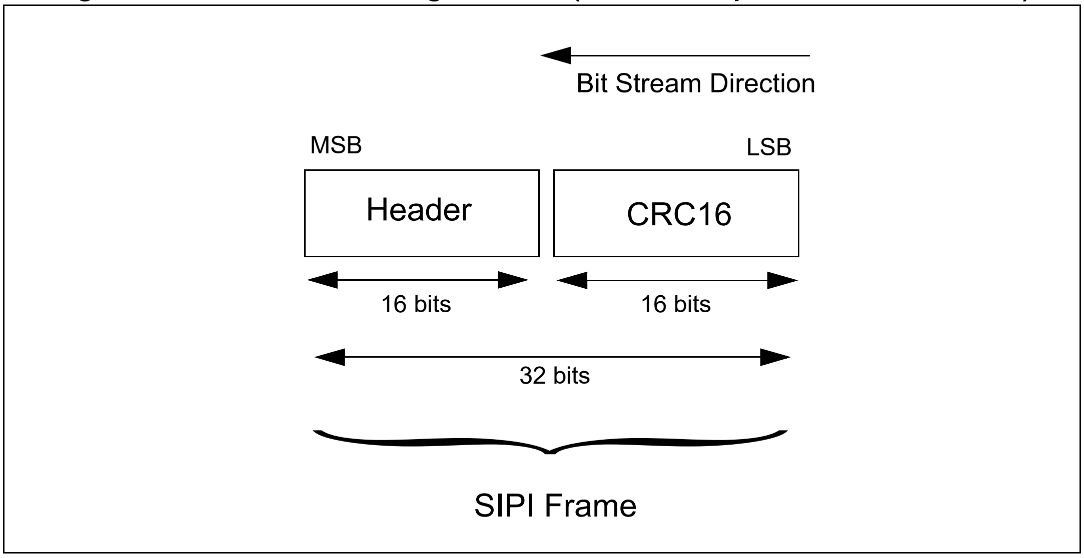
Bit[0]:

- *0 – half-word 1 enabled (MSB)*
- *1 – half-word 0 enabled (LSB)*

65.7.4 Write acknowledge transfer

A write acknowledge transfer contains only header and CRC bits (refer to *Figure 1327*). The CRC bits are calculated on the header field.

Figure 1327. Write acknowledge transfer (LFAST encapsulation is not shown)

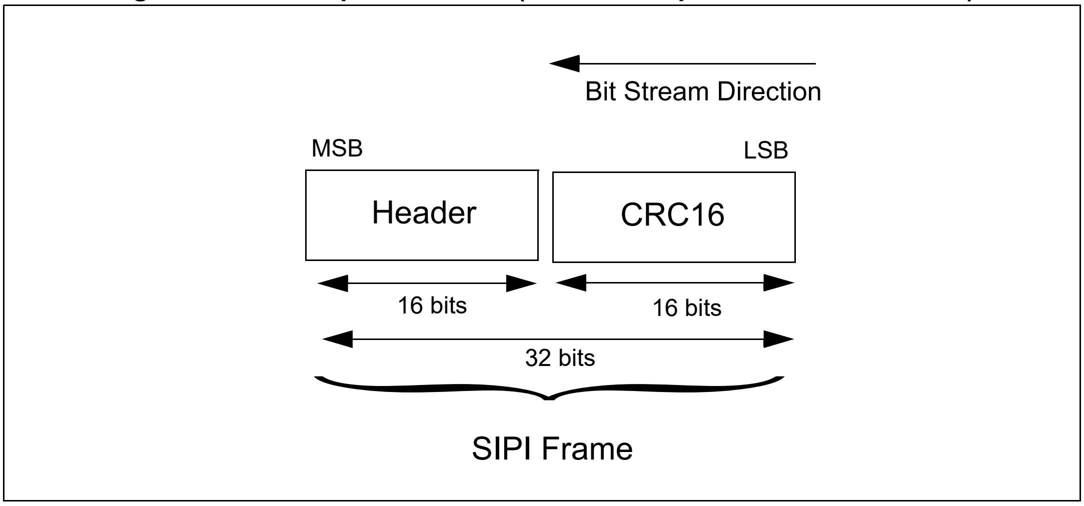


65.7.5 ID request response

65.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in *Figure 1328*.

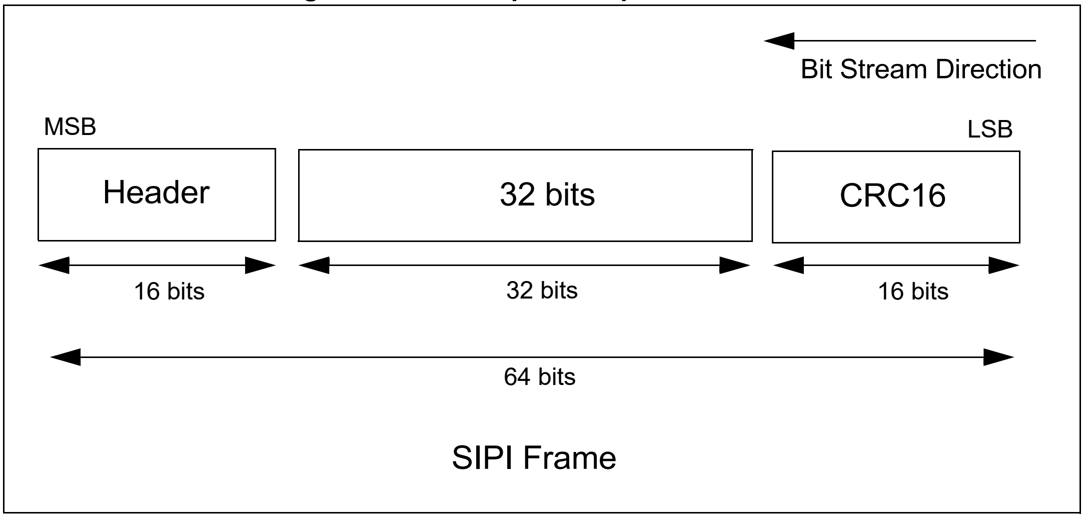
Figure 1328. ID request transfer (LFAST encapsulation is not shown)



65.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in *Figure 1329*.

Figure 1329. ID request response transfer



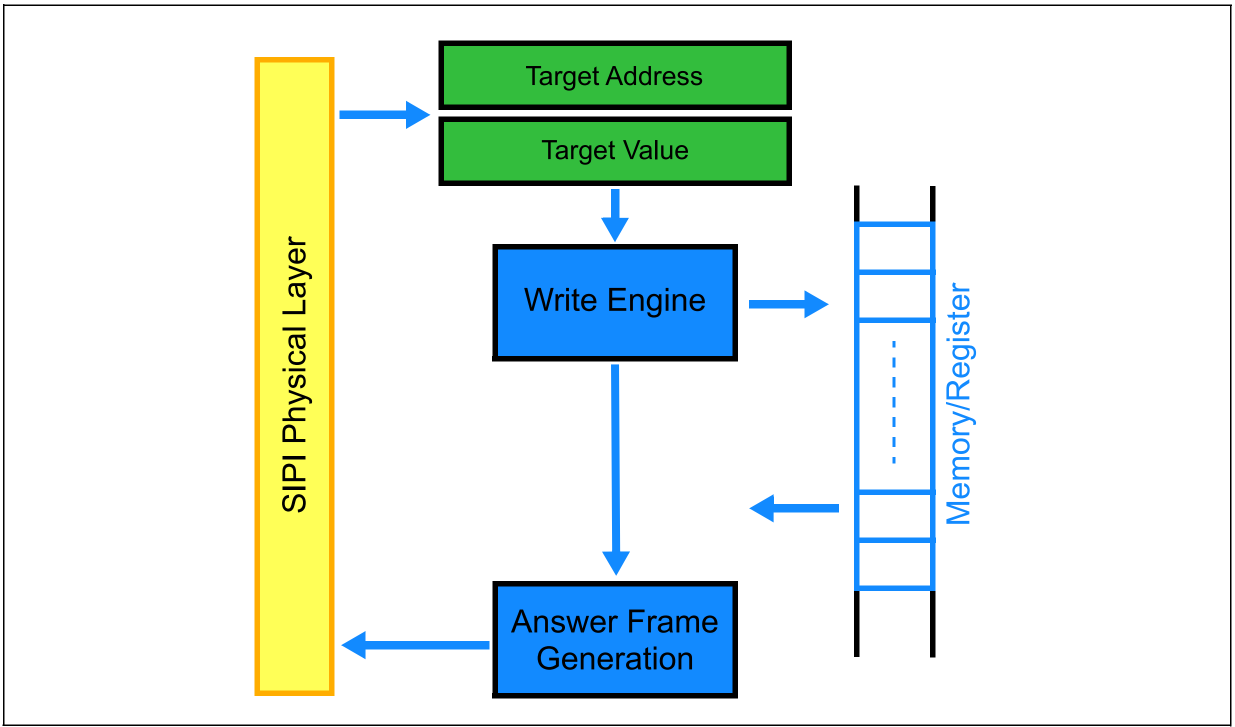
Note: The ID request response contains the value of the CHIP ID in place of data.

65.8 Transfer API and flow charts

On a single write transfer request reception (refer to *Figure 1330*):

1. Debug SIPI will place the address, data and control information on its AHB master interface.
2. When the process in completed, the Debug SIPI will generate an acknowledge frame and send it back to the LFAST.

Figure 1330. SIPI single register write API – Flow chart



On multiple write transfer request reception:

1. Debug SIPI will place the address, data and control information on its AHB Master Interface.
2. When the process in completed, Debug SIPI will generate an acknowledge frame and send it back to the LFAST.

On single read transfer request reception:

1. Debug SIPI will place the address and control information on its AHB Master Interface.
2. When the process in completed, Debug SIPI will send read response back to LFAST.

On multiple read transfer request reception:

1. Debug SIPI then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

65.9 CRC calculation

Example 1

If header is 0xAABB, address is 0x1122_3344 and data is 0xCCDD_EEFF. Then CRC calculation will take place as follows:

1. The CRC seed is initialized by 0xFFFF_FFFF.
2. All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).
3. So the header will be sent as 0xDD55_0000, byte_enable = 0xC.
4. Address will be sent as 0x22CC_4488, byte_enable = 0xF.
5. Data will be sent as 0xFF77_BB33, byte enable = 0xF.

65.10 Memory map and register description

Table 1313. SIPI memory map

Offset from SIPI base (hex)	Register	Section/Page
0x0000–0x009B	Reserved	
0x009C	SIPI Module Configuration Register (SIPI_MCR)	Section 65.10.1.1
0x00A0	SIPI Status Register (SIPI_SR)	Section 65.10.1.2
0x00A4–0x00B0	Reserved	

65.10.1 Register descriptions

65.10.1.1 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a 32-bit global configuration register of the SIPI.

Offset: 0x009C Access: User read/write

	0	11	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZ	0	HALT	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	1	INIT(1)	MOEN	SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

1. Can only be written when SIPI_MCR[MOEN] = 1.

Figure 1331. SIPI Module Configuration Register (SIPI_MCR)

Table 1314. SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. 0 No Freeze mode request. 1 Enters Freeze mode if FRZ bit is asserted.
29 INIT	Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits cannot be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, future SIPI Tx transfers are disabled.
31 SR	Soft Reset Setting this bit clears all status and error registers and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete. Statical configuration, data and address registers remain unchanged.

65.10.1.2 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Offset: 0x00A0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZACK	LPMACK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1332. SIPI Status Register (SIPI_SR)

Table 1315. SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited. 0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge This read-only bit indicates that SIPI is in Disable Mode. Disable mode can not be entered until all current transmission or reception processes have finished. The CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI not in any low power modes 1 SIPI is in Disable mode

66 LVDS Fast Asynchronous Serial Transmission – High Speed Debug

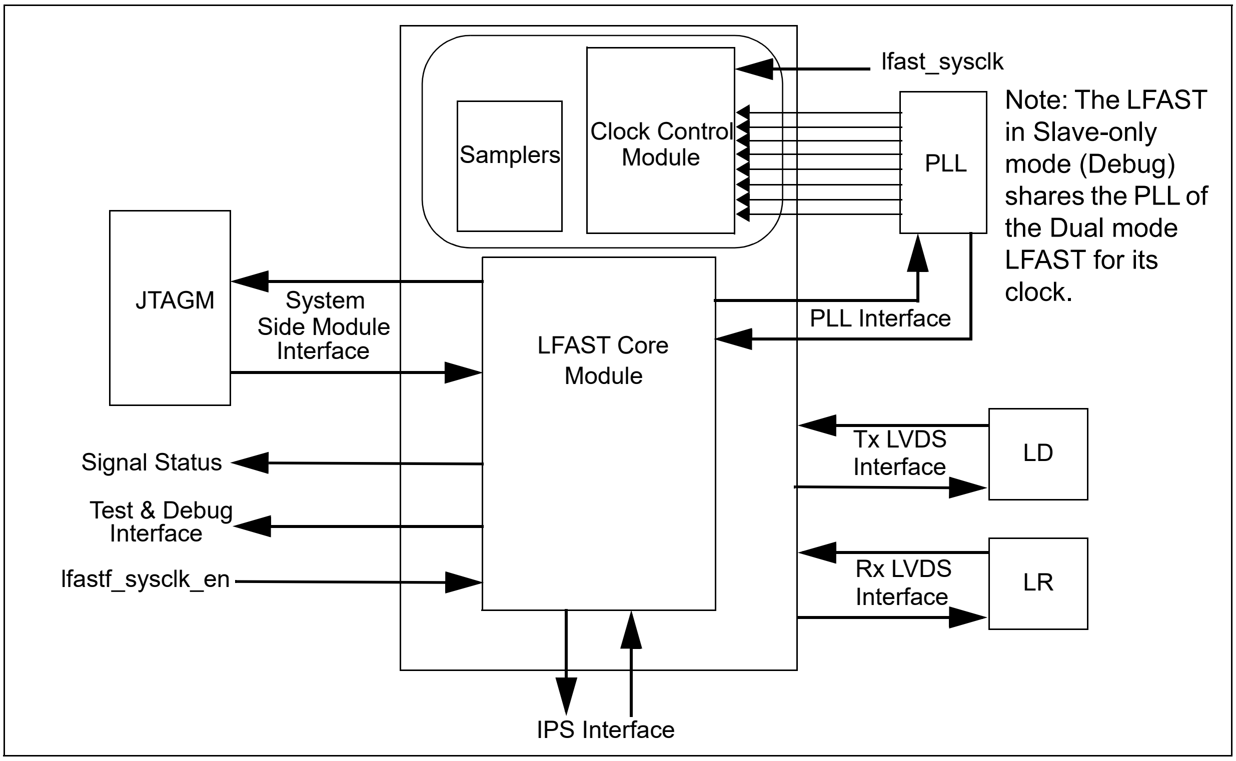
66.1 Introduction

This chapter describes the specifications of the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in slave only mode enabling high speed debug communications through the JTAG port.

66.2 Block diagram

Figure 1333 depicts LFAST interaction with other modules on the device.

Figure 1333. LFAST block diagram



66.3 External signals

LFAST is a six pin interface with the following signals after enabling high speed debug mode:

- lfast_sysclk_en
  - Interface enable signal in LFAST Slave Only mode
- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

66.3.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low data rate, and the high data rate is in the Data Sheet.

66.4 LFAST frame structure

LFAST frames are transmitted and received across a serial interface using a packeted asynchronous communication format.

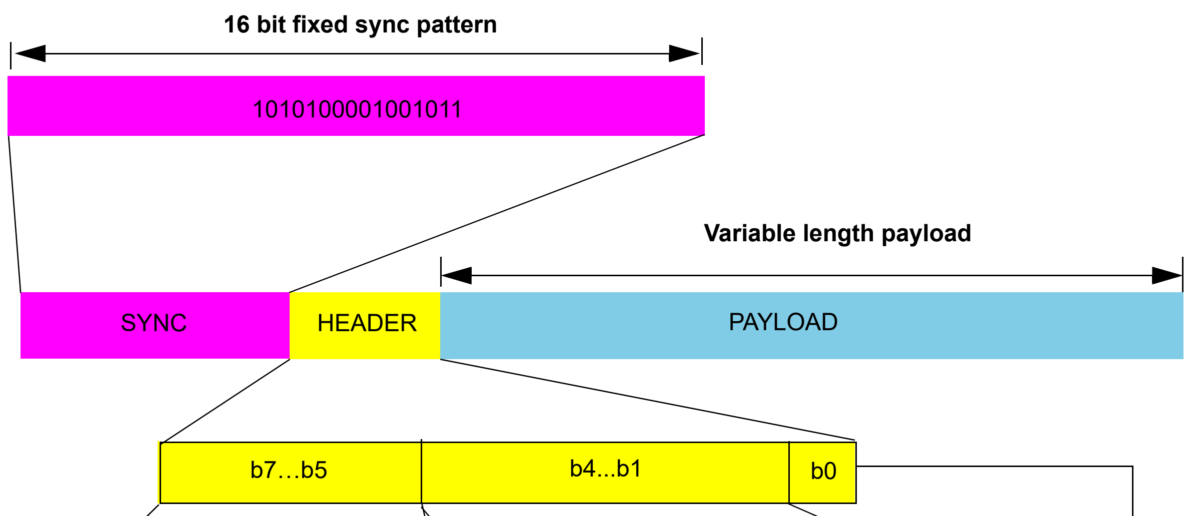
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length. A sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See *Figure 1334* for details of the LFAST frame.

Figure 1334. LFAST frame structure



b7…b5	Payload size (bits)	Frame size (bits)
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	Not Supported	Not Supported

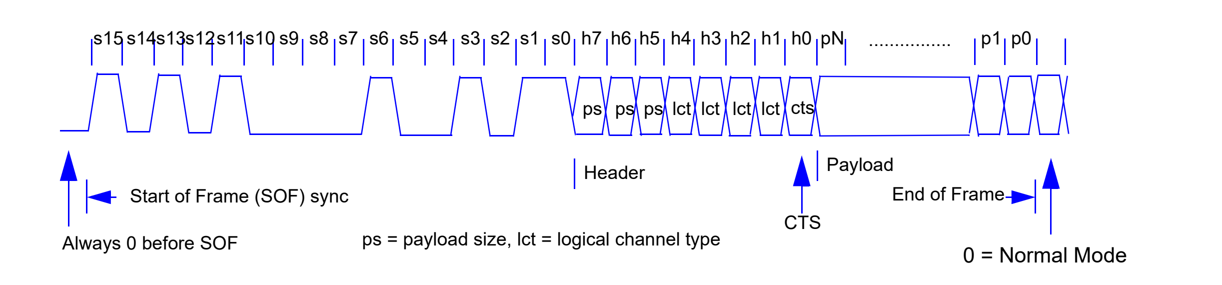
b7…b5 bits of the header

b4…b1	Channel type	Use case
0000	Interface Control	LFAST Master sends ICLC. LFAST Slave sends Ping response
0001	Unsolicited status	Both LFAST master and slave link.
0100	DATA Channel A	For Data transfer by LFAST master and slave
0101	DATA Channel B	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	—

b4…b1 bits of the header

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

Figure 1335. Serial frame structure



The frame structure is shown in *Figure 1335* and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto-correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size according to *Table 1316*.

Table 1316. Header payload sizes

b7-b5 (bin)	Payload size (bits)	Frame size (bits)
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	288	312
111	—	—

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type.
- **Header - (b0):** CTS on the both LFAST master and slave devices. In slave only mode the CTS field will always be 1.
- **Payload: Content dependent upon frame type.**
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode).

66.5 Features

- Supports slave only mode
- Supports asynchronous data transfer rates up to the maximum data rated shown in the product Datasheet
- Transmits and receives data and unsolicited frames
- Receives ICLC frames
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes
- Supports flow control using sliding window protocol
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits
- Supports PLL configuration (such as feedback loop divider) through registers
- Supports LVDS configuration through registers
- Supports multiple loopback modes for checking the physical interface
- Supports automatic ping response generation
- Supports for detection of unsupported channel number and unsupported payload size

66.6 Memory map and register definition

All registers are 32-bit wide.

66.6.1 Memory map

Table 1317. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0000	LFAST Mode Configuration Register (MCR)	R/W	0x0000_0010	*Section 66.6.2.1*
0004	LFAST Speed Control Register (SCR)	R/W	0x0001_0000	*Section 66.6.2.2*
0008	LFAST Correlator Control Register (COCR)	R/W	0x0000_000E	*Section 66.6.2.3*
000C	LFAST Test Mode Control Register (TMCR)	R/W	0x0000_0000	*Section 66.6.2.4*
0010	LFAST Auto Loopback Control Register (ALCR)	R/W	0x0000_0000	*Section 66.6.2.5*
0014	LFAST Rate Change Delay Control Register (RCDCR)	R/W	0x000F_0000	*Section 66.6.2.6*
0018	LFAST Wakeup Delay Control Register (SLCR)	R/W	0x1201_5F02	*Section 66.6.2.7*
001C	Reserved			
0020	LFAST Ping Control Register (PICR)	R/W	0x0000_80CA	*Section 66.6.2.8*
0024–002C	Reserved			
0030–0038	Reserved – Access will not generate errors			

Table 1317. LFAST memory map (continued)

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
003C	LFAST PLL Control Register (PLLCR)	R/W	0x0000_005C	Section 66.6.2.9
0040	LFAST LVDS Control Register (LCR)	R/W	0x0000_5024	Section 66.6.2.10
0044	LFAST Unsolicited Tx Control Register (UNSTCR)	R/W	0x0000_0000	Section 66.6.2.11
0048	LFAST Unsolicited Tx Data Registers 8 (UNSTDR8)	R/W	0x0000_0000	Section 66.6.2.12
004C	LFAST Unsolicited Tx Data Registers 7 (UNSTDR7)	R/W	0x0000_0000	Section 66.6.2.12
0050	LFAST Unsolicited Tx Data Registers 6 (UNSTDR6)	R/W	0x0000_0000	Section 66.6.2.12
0054	LFAST Unsolicited Tx Data Registers 5 (UNSTDR5)	R/W	0x0000_0000	Section 66.6.2.12
0058	LFAST Unsolicited Tx Data Registers 4 (UNSTDR4)	R/W	0x0000_0000	Section 66.6.2.12
005C	LFAST Unsolicited Tx Data Registers 3 (UNSTDR3)	R/W	0x0000_0000	Section 66.6.2.12
0060	LFAST Unsolicited Tx Data Registers 2 (UNSTDR2)	R/W	0x0000_0000	Section 66.6.2.12
0064	LFAST Unsolicited Tx Data Registers 1 (UNSTDR1)	R/W	0x0000_0000	Section 66.6.2.12
0068	LFAST Unsolicited Tx Data Registers 0 (UNSTDR0)	R/W	0x0000_0000	Section 66.6.2.12
006C–007C	Reserved			
0080	LFAST Global Status Register (GSR)	R	0x0000_0000	Section 66.6.2.13
0084–0090	Reserved			
0094	LFAST Data Frame Status Register (DFSR)	R	0x0000_0000	Section 66.6.2.14
0098	LFAST Tx Interrupt Status Register (TISR)	R/W	0x0000_0000	Section 66.6.2.15
009C	LFAST Rx Interrupt Status Register (RISR)	R/W	0x0000_0000	Section 66.6.2.16
00A0	LFAST Rx ICLC Interrupt Status Register (RIISR)	R/W	0x0000_0000	Section 66.6.2.17
00A4	LFAST PLL and LVDS Status Register (PLLLSR)	R	0x0002_0003	Section 66.6.2.18
00A8	LFAST Unsolicited Rx Status Register (UNSRSR)	R/W	0x0000_0000	Section 66.6.2.19
00AC	LFAST Unsolicited Rx Data Register 8 (UNSRDR8)	R	0x0000_0000	Section 66.6.2.20
00B0	LFAST Unsolicited Rx Data Register 7 (UNSRDR7)	R	0x0000_0000	Section 66.6.2.20
00B4	LFAST Unsolicited Rx Data Register 6 (UNSRDR6)	R	0x0000_0000	Section 66.6.2.20
00B8	LFAST Unsolicited Rx Data Register 5 (UNSRDR5)	R	0x0000_0000	Section 66.6.2.20
00BC	LFAST Unsolicited Rx Data Register 4 (UNSRDR4)	R	0x0000_0000	Section 66.6.2.20
00C0	LFAST Unsolicited Rx Data Register 3 (UNSRDR3)	R	0x0000_0000	Section 66.6.2.20
00C4	LFAST Unsolicited Rx Data Register 2 (UNSRDR2)	R	0x0000_0000	Section 66.6.2.20

Table 1317. LFAST memory map (continued)

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
00C8	LFAST Unsolicited Rx Data Register 1 (UNSRDR1)	R	0x0000_0000	Section 66.6.2.20
00CC	LFAST Unsolicited Rx Data Register 0 (UNSRDR0)	R	0x0000_0000	Section 66.6.2.20

66.6.2 Register descriptions

66.6.2.1 Mode Configuration Register (MCR)

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0	0	0	0	0	0	0	0	TXARBD	0	0	DRFRST(1)	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1336. Mode Configuration Register (MCR)

1. Set by user software and cleared by system hardware.

Table 1318. MCR field descriptions

Field	Description
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode Refer to *Section 66.10.2: Slow speed clock* for more detailed description. 0 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.

Table 1318. MCR field descriptions (continued)

Field	Description
17 RXEN	LFAST Receiver Enable This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS LR. 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface is not disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be set and cleared by LFAST slave hardware on reception of an ICLC command frame. 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 Not allowed. Do not set when in slave only mode.
27 TXARBD	Tx Arbiter Disable This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
30 DRFRST	LFAST Soft Reset This bit is automatically cleared after reset. 0 No soft reset. 1 Soft reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

66.6.2.2 Speed Control Register (SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Offset: 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DRMD(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RDR	0	0	0	0	0	0	0	TDR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 0.

Figure 1337. Speed Control Register (SCR)

Table 1319. SCR field descriptions

Field	Description
15 DRMD	Data Rate Controller mode Defines the mode set for LFAST slave device by the software or the LFAST master. This bit should not be modified by the software and left as SCR[DRMD] =1.
23 RDR	Receiver Data Rate This bit defines the receiver data rate. This bit should not be modified by the software.
31 TDR	Transmit Data Rate This bit defines the transmitter data rate. This bit should not be modified by the software.

66.6.2.3 Correlator Control Register (COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMPSEL(1)								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CORRTH(1)			PHSSEL(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

1. Only writable only when MCR[RXEN] = 0.

Figure 1338. Correlator Control Register (COCR)

Table 1320. COCR field descriptions

Field	Description
0:7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection) Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. 0x00 Sampler Data Path selected by correlator 0x01 Sampler Data Path 0 selected 0x02 Sampler Data Path 1 selected 0x04 Sampler Data Path 2 selected 0x08 Sampler Data Path 3 selected 0x10 Sampler Data Path 4 selected 0x20 Sampler Data Path 5 selected 0x40 Sampler Data Path 6 selected others Sampler Data Path 7 selected
28:30 CORRTH	Correlator threshold level. Defines the correlation threshold level. 000 9 Bits of correlation 001 10 Bits of correlation …. 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. 0 8 phases 1 4 phases

66.6.2.4 Test Mode Control Register (TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes

Offset: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	CLKTST	LPON	0	0	0	0	0	LPMOD[(1)]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	LPFRMTH[(1)]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Figure 1339. Test Mode Control Register (TMCR)

Table 1321. TMCR field descriptions

Field	Description
6 CLKTST	Clock Test mode The software can define when the clock test mode is enabled. This bit can also be set and cleared by hardware on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable The software can define when the loopback logic is enabled. This bit can also be written by LFAST slave hardware on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Loopback mode is enabled. 0 Loopback mode is disabled.
13:15 LPMOD	Loopback mode Defines the type of loopback mode enabled. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation Others - Reserved (should not be written by the software)
16:31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0x0000: Reserved (Not to be used) 0x0001: Check 1 frame have correct sync, header and payload. … 0xFFFE: Check 65534 frames have correct sync, header and payload. 0xFFFF: Check 65535 frames have correct sync, header and payload.

66.6.2.5 Auto Loopback Control Register (ALCR)

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LPCNTEN(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	LPFMCNT(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Figure 1340. Auto Loopback Control Register (ALCR)

Table 1322. ALCR field descriptions

Field	Description
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable Enables fixed number of auto predefined loopback frame transmission. 0 Infinite predefined loopback frame transmission enabled 1 Fixed count of predefined loopback frame transmission enabled
16:31 LPFMCNT	Auto Loopback Frame Transmission Count Defines the number of predefined auto loopback frames to be sent. The predefined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0x0000: Reserved (Not to be used) 0x0001: Send 1 frame with correct sync, header and payload. ... 0xFFFE: Send 65534 frames with correct sync, header and payload. 0xFFFF: Send 65535 frames with correct sync, header and payload.

66.6.2.6 Rate Change Delay Control Register (RCDCR)

Offset: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DRCNT[1]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 1.

Figure 1341. Rate Change Delay Control Register (RCDCR)

Table 1323. RCDCR field description

Field	Description
12:15 DRCNT	Data Rate Controller Counter Value Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. 0x0: 0 cycles 0x1: 1 cycles ... 0xE: 14 cycles 0xF: 15 cycles

66.6.2.7 Wakeup Delay Control Register (SLCR)

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HSCNT[1]								0	0	0	0	LSCNT[1]			
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HWKCNT[1]								0	0	0	0	LWKCNT[1]			
W																
Reset	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

1. Only writable when MCR[DRFEN] = 0.

Figure 1342. Wakeup Delay Control Register (SLCR)

Table 1324. SLCR field descriptions

Field	Description
0:7 HSCNT	High Speed Sleep mode Exit Time Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. 0x00: cycle 0x01: 1 cycle … 0x12: 18 cycles (200 ns + 8 cycles of High speed clock) … 0xFE: 254 cycles 0xFF: 255 cycles
12:15 LSCNT	Low Speed Sleep mode Exit Time Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. 0x0: 0 cycle 0x1: 1 cycle (200ns + 1 cycle of Low speed clock) … 0xE: 14 cycles 0xF: 15 cycles
16:23 HWKCNT	Wake Up time for the LD Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. 0x00: 0 cycles 0x01: 1 cycle … 0x5F: 95 cycles (1.18 us) … 0xFF: 255 cycles Maximum
28:31 LWKCNT	Wake Up time for the LD Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. 0x0: 0 cycle 0x1: 1 cycle 0x2: 2 cycles (1.18 us) … 0xE: 14 cycles 0xF: 15 cycles

66.6.2.8 Ping Control Register (PICR)

Offset: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PNGREQ(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PNGAUTO(2)	0	0	0	0	0	0	0	PNGPYLD(2)							
W																
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

1. Set by user software and cleared by system hardware.
2. Only writable when MCR[DRFEN] = 0.

Figure 1343. Ping Control Register (PICR)

Table 1325. PICR field descriptions

Field	Description
15 PNGREQ	Ping Response Frame Request This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. This field can only be written when MCR[DRFEN] = 1. 1 Ping response frame transmission request is queued. 0 No pending Ping response frame transmission request.
16 PNGAUTO	Ping Response Enable Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master This bit should not be modified by the software.
24:31 PNGPYLD	Defines the LFAST slaves ping reply frame payload content.

66.6.2.9 PLL Control Register (PLLCR)

This register exists only when LFAST controls an external PLL.

Offset: 0x003C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPTMOD			0	0	0	0	0	0	0	0	0	0	0	SWPOFF	SWPON
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REFINV	LPCFG		0	0	PLCKCW		FDIVEN	FBDIV(1)						PREDIV(1)	
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0

Figure 1344. PLL Control Register (PLLCR)

1. Only writable when MCR[DRFEN] = 0.

Table 1326. PLLCR field descriptions

Field	Description
0:2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 1 010 Force Vctrl 011 Charge Pump Up 100 Charge Pump Up Internal Test 101 Charge Pump Idle 110 Charge Pump Down 111 Closed Loop 2
14 SWPOFF	Software signal to turn OFF the PLL This bit should not be modified by the software.
15 SWPON	Software signal to turn ON the PLL This bit should not be modified by the software.
16 REFINV	Inverts reference clock edge to PFD If System PLL PFD using same reference clock, enabling this feature will minimize noise injection crosstalk via the substrate. 0 Not inverted 1 Invert
17:18 LPCFG	PLL Loop Optimization. Adjusts charge pump bias current. Higher current increases PLL bandwidth, decreasing jitter while degrading stability. 00 1 x IBASE 01 0.5 x IBASE 10 1.5 x IBASE 11 2 x IBASE

Table 1326. PLLCR field descriptions (continued)

Field	Description
21:22 PLCKCW	PLL Lock Ready Count Width Defines the number of cycles PLL waits for before asserting lock_ready flag High. 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24:29 FBDIV	Feedback Division factor for VCO output clock 0x00: No clock output 0x01: Reserved … 0x0A: Reserved 0x0B: Divide by 11 (11.5, if FDIVEN = 1) … 0x1F: Divide by 31 (31.5, if FDIVEN = 1) 0x20: Divide by 32 (32.5, if FDIVEN = 1) 0x21: Reserved … 0x3F: Reserved
30:31 PREDIV	Division factor for PLL Reference Clock input 00 Direct clock passed 01 Divide by 2 10 Divide by 3 11 Divide by 4

66.6.2.10 LVDS Control Register (LCR)

Offset: 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SWWKLD[(1)]	SWSLPLD[(1)]	SWWKLR[(1)]	SWSLPLR[(1)]	SWOFFLD	SWONLD	SWOFFLR	SWONLR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0	0	0	0	0	LVRXOP_TR	Reserved	LVRXOP_BR	LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

1. Set by user software and cleared by system hardware.

Figure 1345. LVDS Control Register (LCR)

Table 1327. LCR field description

Field	Description
8 SWWKLD	Software signal to take LVDS LD out of Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep).
9 SWSLPLD	Software signal to put LVDS LD into Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be put in sleep.
10 SWWKLR	Software signal to take LVDS LR out of Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep).
11 SWSLPLR	Software signal to put LVDS LR into Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up).
12 SWOFFLD	Software signal to turn OFF the LVDS LD This bit can be changed only when DRFEN is high. This bit should not be modified by the software.

Table 1327. LCR field description (continued)

Field	Description
13 SWONLD	Software signal to turn ON the LVDS LD This bit can be changed only when DRFEN is high. This bit should not be modified by the software.
14 SWOFFLR	Software signal to turn OFF the LVDS LR This bit can be changed only when DRFEN is high. This bit should not be modified by the software.
15 SWONLR	Software signal to turn ON the LVDS LR This bit can be changed only when DRFEN is high. This bit should not be modified by the software.
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode.
17 LVTXOE	LVDS LD output buffer enable 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad.
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only). 0 Disable on-chip LFAST receiver termination 1 Enable on-chip LFAST receiver termination
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only). 0 Use for LFAST receiver baud rates less than maximum baud rate 1 Required for LFAST receiver maximum baud rate
29 LVTXOP	Control signal for LFAST and Micro-Second Bus selection 0 Micro-Second Bus selection 1 LFAST Bus selection

Table 1327. LCR field description (continued)

Field	Description
30 LVCKSS	LVDS Clock Sync Select This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 Normal clock used to sample the LVDS data 1 Adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

66.6.2.11 Unsolicited Tx Control Register (UNSTCR)

Offset: 0x0044 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	USNDRQ[1]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	UNSHDR[2]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Set by user software and cleared by system hardware.
2. Only writable when UNSTCR[USNDRQ] = 0.

Figure 1346. Unsolicited Tx Control Register (UNSTCR)

Table 1328. UNSTCR field descriptions

Field	Description
15 USNDRQ	Tx Unsolicited send request. This field can only be written when MCR[DRFEN] = 1. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
25:31 UNSHDR	Tx Unsolicited message header

66.6.2.12 LFAST Unsolicited Tx Data Registers (UNSTDR*n*)

Offset: 0x0048 (UNSTDR8)
0x004C (UNSTDR7)
0x0050 (UNSTDR6)
0x0054 (UNSTDR5)
0x0058 (UNSTDR4)
0x005C(UNSTDR3)
0x0060 (UNSTDR2)
0x0064 (UNSTDR1)
0x0068 (UNSTDR0)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	UNTXD(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	UNTXD(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0.

Figure 1347. Unsolicited Tx Data Registers (UNSTDR8–UNSTDR0)

Table 1329. UNSTDR8–UNSTDR0 field descriptions

Field	Description
0:31 UNTXD	Unsolicited Transmit Data 8–0 This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

66.6.2.13 Global Status Register (GSR)

Offset: 0x0080 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0	0	0	0	0	0	0	0	0	0	0	0	LRMD	LDSM	DRSM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LPTXDN	LPFPDV	LPCPDV	LPCHDV	LPCSDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1348. Global Status Register (GSR)

Table 1330. GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module Slave only mode 1 LFAST Module in dual mode
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active. 0 Rx Controller is in Idle state. 1 Rx Controller is active.
14 LDSM	Transmit Interface Data Rate Status Indicates the current speed rate of the Tx controller. 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status Indicates the current speed rate of the Rx controller. 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
27 LPTXDN	Auto loopback frame transmission count reached The Count of frame is defined by *Auto Loopback Control Register (ALCR)* LPFMCNT. 0 Auto loopback frame transmission count not reached 1 Auto loopback frame transmission count reached
28 LPFPDV	Loopback frame pass threshold reached 0 Pass frame threshold not reached 1 Pass frame threshold achieved

Table 1330. GSR field descriptions (continued)

Field	Description
29 LPCPDV	Valid payload received during loopback check mode Indicates whether the Loop back frame received payload is 0xCB. 0 Payload received is not 0xCB. 1 Payload received is 0xCB.
30 LPCHDV	Valid header received during loopback check mode Indicates whether the Loop back frame received header is 0x13. 0 Header received is not 0x13. 1 Header received is 0x13.
31 LPCSDV	Valid synchronization received 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

66.6.2.14 Data frame Status Register (DFSR)

Offset: 0x0094 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXDCNT						0	0	0	0	0	RXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			TXDCNT						0	0	0	0	0	TXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1349. Data frame Status Register (DFSR)

Table 1331. DFSR field descriptions

Field	Description
2:7 RXDCNT	Unread Rx Frame Data Count Indicates the number of unread data stored in the Rx Data FIFO.
13:15 RXFCNT	Unread Rx Frame Count Indicates the number of unread data frames stored in the Rx Data FIFO.
18:23 TXDCNT	Unread Tx Frame Data Count Indicates the number of unread data stored in the Tx Data FIFO.
29:31 TXFCNT	Unread Tx Frame Count Count of pending Data Frames programmed by System Side Module.

66.6.2.15 Tx Interrupt Status Register (TISR)

Offset: 0x0098 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1350. Tx Interrupt Status Register (TISR)

Table 1332. TISR field descriptions

Field	Description
14 TXIEF	Tx Data Interface not enabled Tx Data Interface not enabled and a frame is ready to be transmitted. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

66.6.2.16 Rx Interrupt Status Register (RISR)

Offset: 0x009C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	RXUOF	0	0	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c			w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1351. Rx Interrupt Status Register (RISR)

Table 1333. RISR field descriptions

Field	Description
8 RXUOF	Unsolicited frame register overflow Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
11 RXUFF	Rx Data FIFO Underflow 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
12 RXOFF	Rx Data FIFO Overflow 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
13 RXSZF	Frame with unsupported frame size received 0 Interrupt event has not occurred. 1 Interrupt event has occurred: On reception of frame with payload size for a frame other than mentioned in *Table 1346*.
14 RXICF	Invalid ICLC code Received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
15 RXLCEF	Invalid Logical Channel Type 0 Interrupt event has not occurred. 1 Interrupt event has occurred: On reception of frame other than mentioned in *Table 1346*.
28 RXCTSF	Frame with CTS bit Low Received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

Table 1333. RISR field descriptions (continued)

Field	Description
29 RXDF	Data frame received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 RXUNSF	Unsolicited Frame received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

66.6.2.17 Rx ICLC Interrupt Status Register (RIISR)

Offset: 0x00A0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1352. Rx ICLC Interrupt Status Register (RIISR)

Table 1334. RIISR field descriptions

Field	Description
18 ICPFF	Ping Frame Response Payload mismatch 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

Table 1334. RIISR field descriptions (continued)

Field	Description
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

66.6.2.18 PLL and LVDS Status Register (PLLLSR)

Offset: 0x00A4 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PLLDIS	PLDCR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LRSLPS	LDSLPS	LDPDS	LRPDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 1353. PLL and LVDS Status Register (PLLLSR)

Table 1335. PLLLSR field descriptions

Field	Description
14 PLLDIS	PLL disable Status When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles. 0 PLL Lock delay counter is not decremented to 0. 1 PLL Lock delay counter is decremented to 0.
28 LRSLPS	This bit indicates the real time status of the LR sleep signal. 0 LR sleep signal is negated. 1 LR sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal. 0 LD power sleep signal is negated. 1 LD power sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal. When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal. When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

66.6.2.19 Unsolicited Rx Status Register (UNSRSR)

Offset: 0x00A8 Access: read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	URXDV	0	0	0	0	0	URPCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1354. Unsolicited Rx Status Register (UNSRSR)

Table 1336. UNSRSR field descriptions

Field	Description
23 URXDV	Unsolicited data valid Indicates a valid frame exists in the Unsolicited Data registers.
29:31 URPCNT	Rx Unsolicited payload Indicates the number of bytes of valid Rx unsolicited data payload present in the *Unsolicited Rx Data Registers (UNSRDRn)*. The bitfield coding is same as b7-b5 in *Table 1316*.

66.6.2.20 Unsolicited Rx Data Registers (UNSRDR*n*)

Offset:
0x00AC (UNSRDR8)
0x00B0 (UNSRDR7)
0x00B4 (UNSRDR6)
0x00B8 (UNSRDR5)
0x00BC (UNSRDR4)
0x00C0 (UNSRDR3)
0x00C4 (UNSRDR2)
0x00C8 (UNSRDR1)
0x00CC (UNSRDR0)

Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UNRXD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1355. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Table 1337. UNSRDR8–UNSRDR0 field descriptions

Field	Description
0:31 UNRXD	Unsolicited Receive Data This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288-bit payload.

66.6.3 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements (Refer to the “Functional safety” chapter for register classification details). All registers of the LFAST module are classified as non-safety relevant.

The MCR[IPGDBG] fulfills the safety requirements for monitoring of the debug freeze signal.

66.7 Functional description

66.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

66.7.1.1 LFAST slave interface startup procedure

1. After POR the RCDCR are programmed according to the LVDS parameter in the device datasheet.
2. The PLLCR[t] is programmed with the configuration parameters of the PLL.
3. The LCR is programmed with the configuration parameters of the LVDS.
4. MCR[DATAEN] field is programmed to select the LFAST mode.
5. Write MCR[TXARBD] = 0 to enable Tx of frames.
6. Write MCR[DRFEN] = 1 to enable LFAST.
7. Write MCR[RXEN] = 1 to enable LFAST Rx.
8. On the assertion of the LFAST interface enable the signals LR power down and LR disable are negated.
9. When an ICLC frame with payload 0x31 is received the hardware writes RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
10. The ping frame is sent when an ICLC frame with payload 0x00 frame is received and the hardware writes RIISR[ICPRF] = 1.

Speed mode change:

11. The PLL starts when an ICLC frame with payload 0x02 is received and the hardware writes RIISR[ICPONF] = 1.
12. The speed of the Tx interface is changed when an ICLC frame with payload 0x80 is received and the hardware writes RIISR[ICTFF] = 1. This will cause SRC[TDR] = 1.
13. The speed of the Rx Interface is changed when an ICLC frame with payload 0x10 is received and the hardware writes RIISR[ICRFF] = 1. This will cause SCR[RDR] = 1.
14. The ping frame is sent when an ICLC frame with payload 0x00 is received and the hardware writes RIISR[ICPRF] = 1.

66.7.2 LFAST Interface enable signal (lfast_sysclk_en)

The lfast_sysclk_en signal comes from the DCI module. It is asserted by the DCI internal state machine when switching from JTAG to LFAST and vice-versa can safely be done. The detailed effects of lfast_sysclk_en on the module are:

LFAST interface enable (lfast_sysclk_en) negation:

t. The PLLCR is shared between the LFAST – Interprocessor Communications and the LFAST – High Speed Debug modules. The programming values for the PLLCR in the LFAST – Interprocessor Communications are also used by the LFAST – High Speed Debug module.

1. If LCR[SWONLD] = 0 the LD LVDS pad is powered down.
2. If LCR[SWONLR] = 0 the LR LVDS pad is powered down.
3. The LFAST slave exits the test mode (for example, loopback mode or clock test mode). Then the software writes MCR[CLKTST] = 0 and MCR[LPON] = 0.
4. The LFAST slave Rx and Tx interfaces are set to Low Speed mode. Then write SCR[TDR] = 0 and SCR[RDR] = 0.
5. The Tx and Rx interface controllers are disabled.
6. The status registers are cleared.

LFAST interface enable (lfast_sysclk_en) assertion:

1. The LD LVDS pad is power up, if all conditions are met:
   – LCR[SWOFFLD] = 0.
   – LCR[SWONLD] = 1.
   – MCR[TXEN] = 1.
   – When frame ICR[ICLCPLD] = 0x31 is received.
2. LR LVDS pad is powered up if all conditions are met:
   – LCR[SWOFFLR] = 0.
   – LCR[SWONLR] = 1.
   – MCR[RXEN] = 1.

66.7.3 Line Receiver (LR)

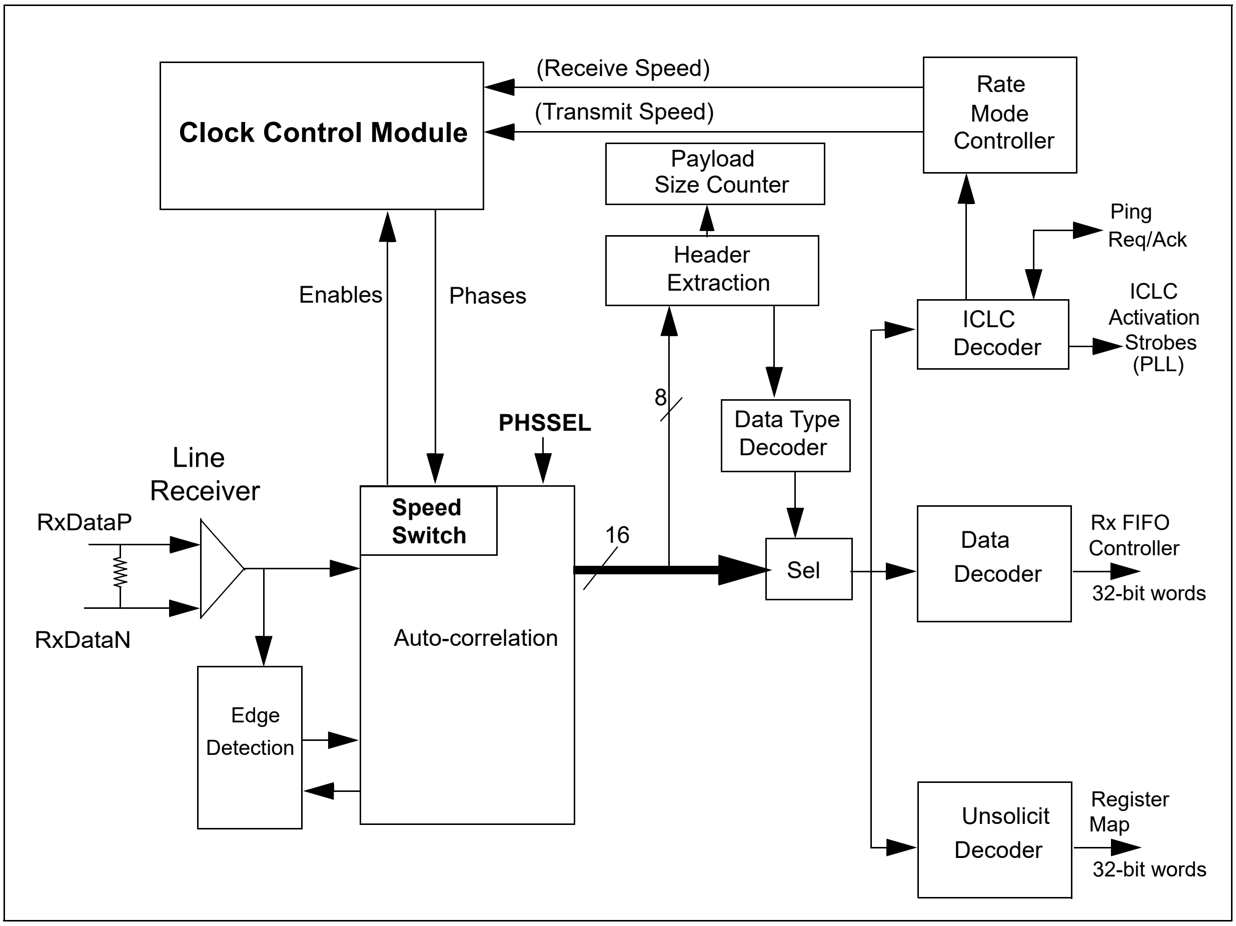
66.7.3.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto-correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 1356 shows the block diagram of Uplink controller.

Figure 1356. Top level receive controller



66.7.3.2 Edge-detection and auto-correlation

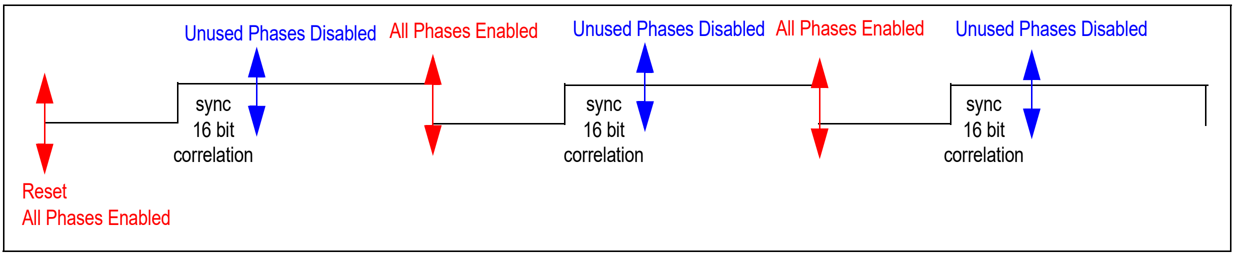
The edge-detection and auto-correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge-detection circuitry is used or not. The edge-detection will be performed first if required before auto-correlation.

66.7.3.2.1 Auto-correlation modes

66.7.3.2.1.1 Hunt correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 1357. Hunt correlation mode



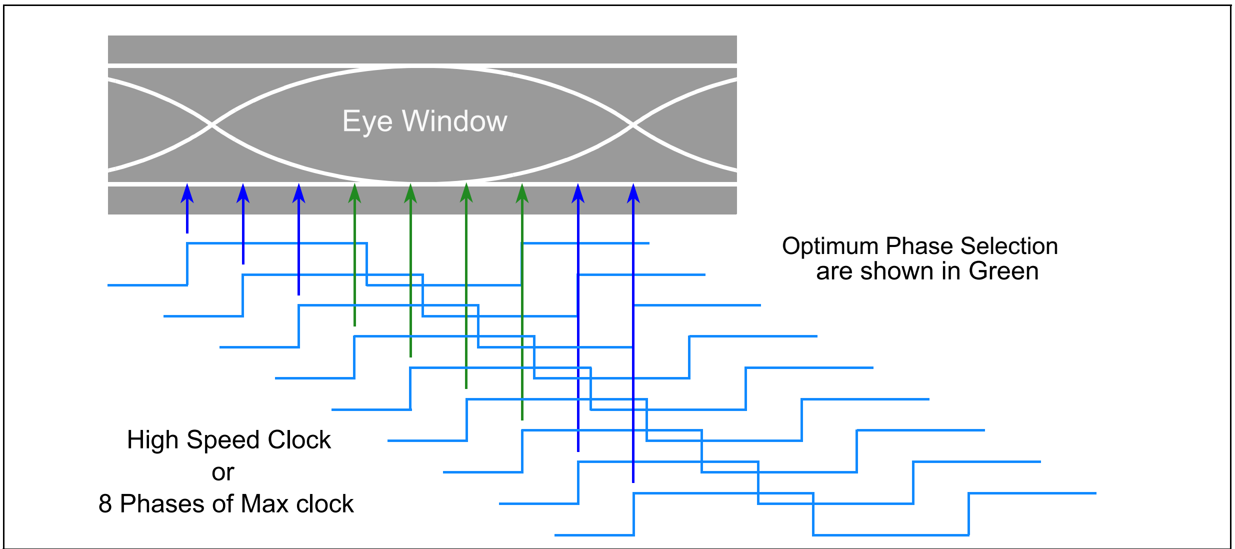
66.7.3.2.2 Edge detection

The edge detector is disabled in Hunt Correlation mode.

66.7.3.2.3 Auto-correlation

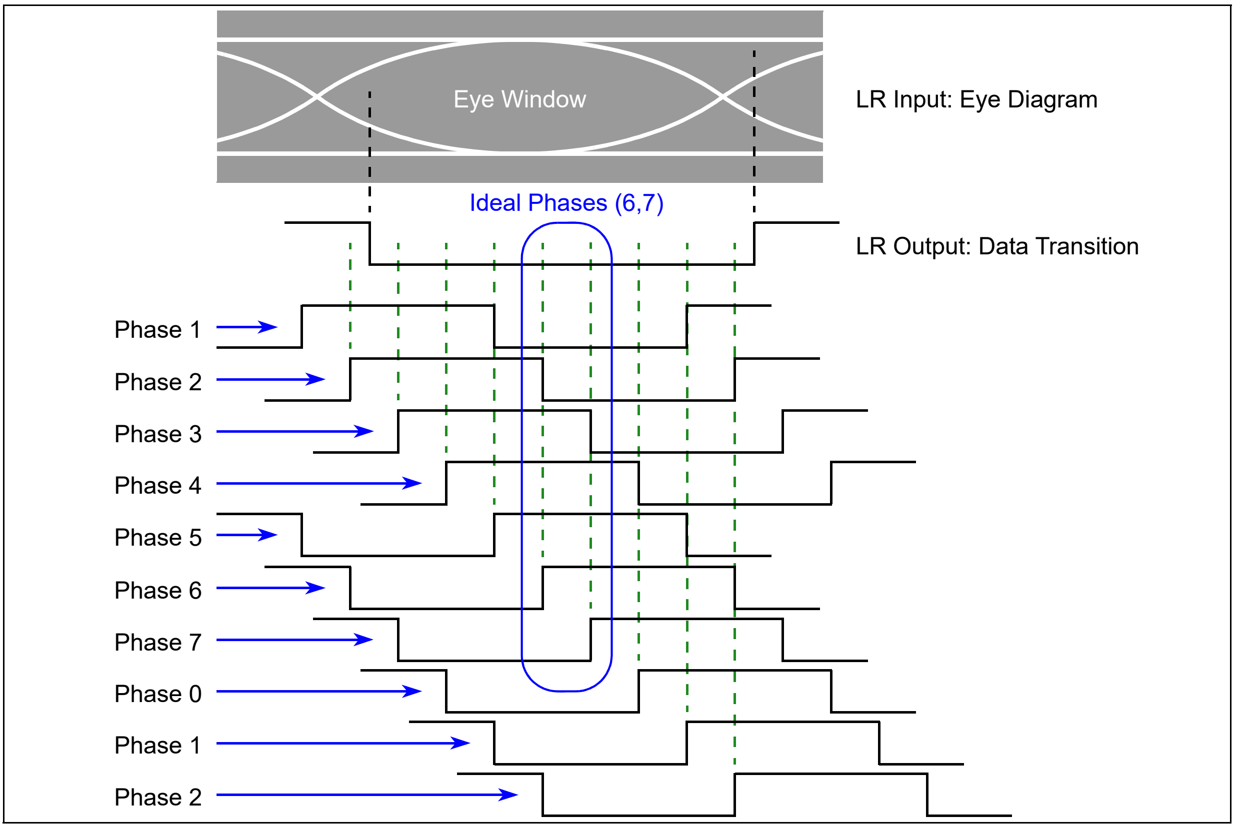
The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto-correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in *Figure 1358*.

Figure 1358. High speed 8 phase selection



The objective of the auto-correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in *Figure 1359*.

Figure 1359. High speed 8 phase clock alignment example



Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in *Figure 1360* and *Figure 1361*.

Figure 1360. High speed 4 phase clock alignment example

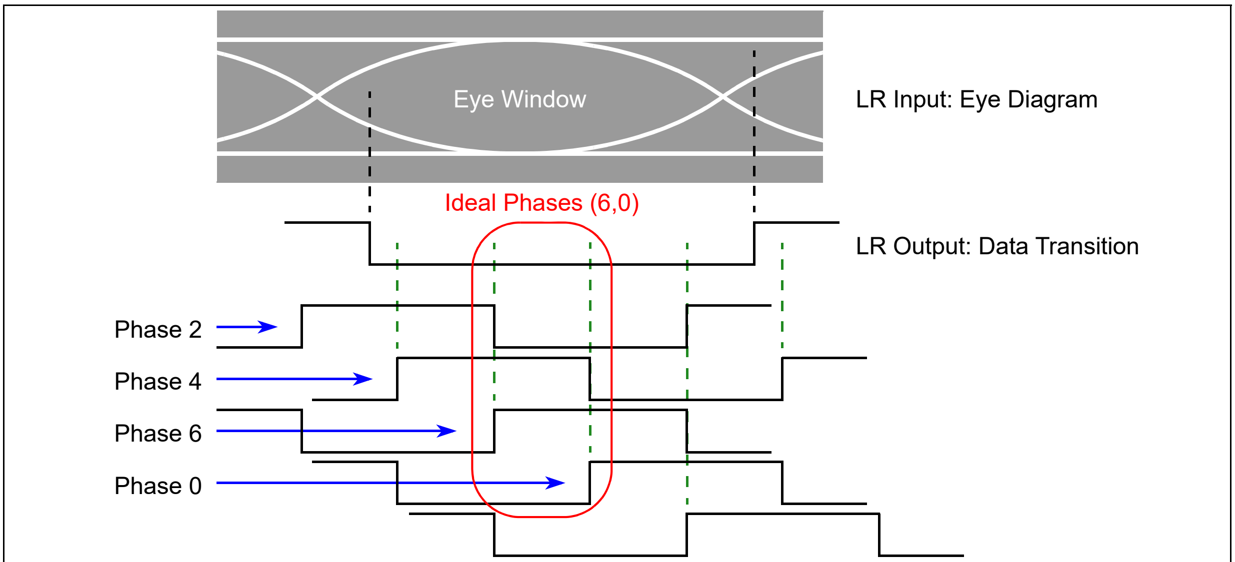
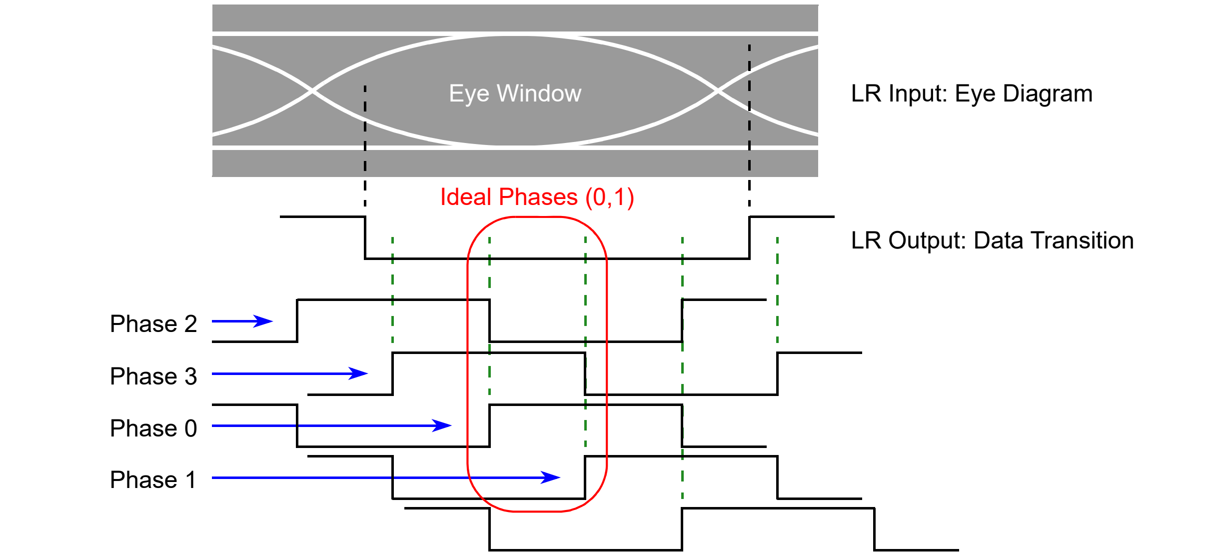


Figure 1361. Low speed 4 phase clock alignment example



The auto-correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High Speed 8 Phases: Samplers 0,1,2,3,4,5,6,7
- High Speed 4 Phases: Samplers 0,2,4,6
- Low Speed 4 Phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and re-samples it to an intermediate phase as shown in *Table 1338*, *Table 1339*, and *Table 1340* before re-sampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 1338. High speed 8 phase selection - sampling procedure

Samplers	Initial Sample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0

Table 1338. High speed 8 phase selection - sampling procedure (continued)

Samplers	Initial Sample	Intermediate Sample	Final Sample
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 1339. High speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See *Table 1339*, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 1340. Low speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See *Table 1340*, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

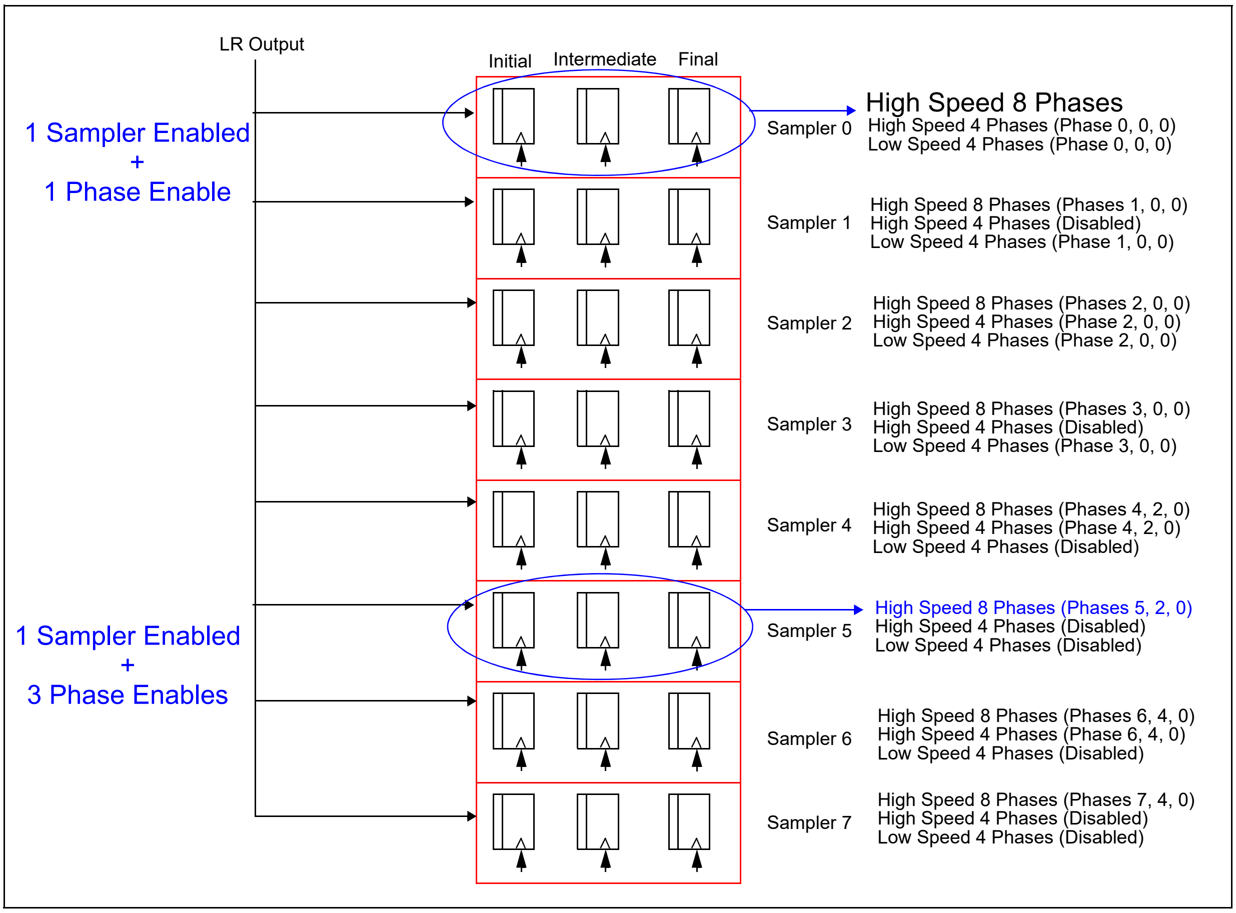
66.7.3.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto-correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in *Figure 1362*.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for instance Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by de-asserting the respective phase enables.

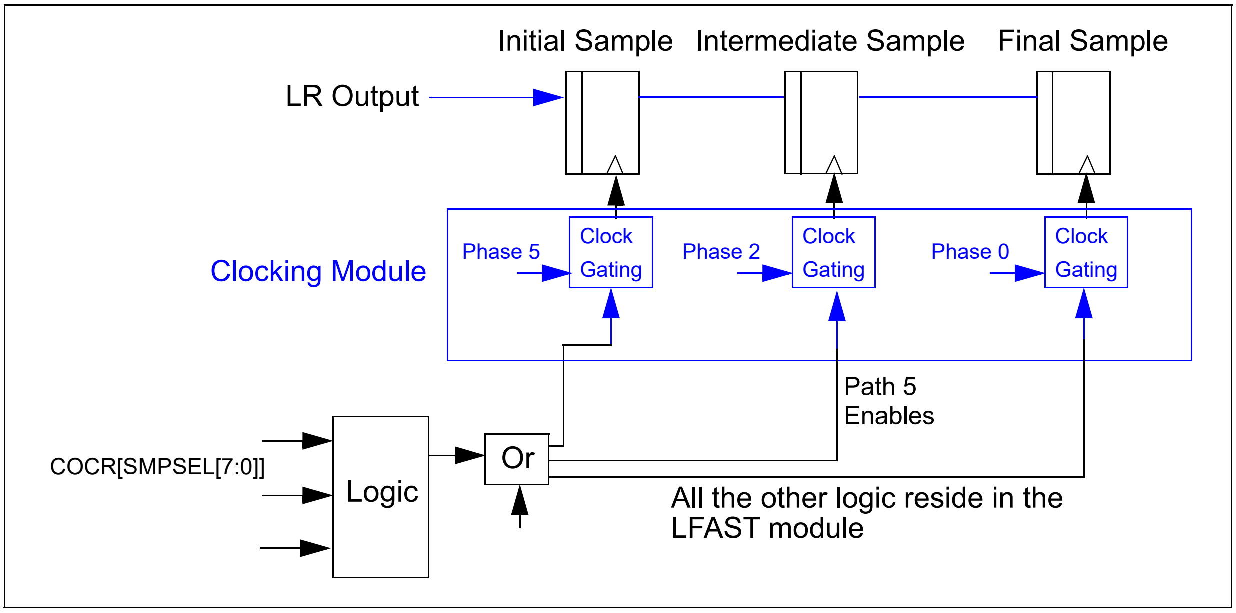
Figure 1362. Samplers, each sampler has 3 registers



In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in *Figure 1363*.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

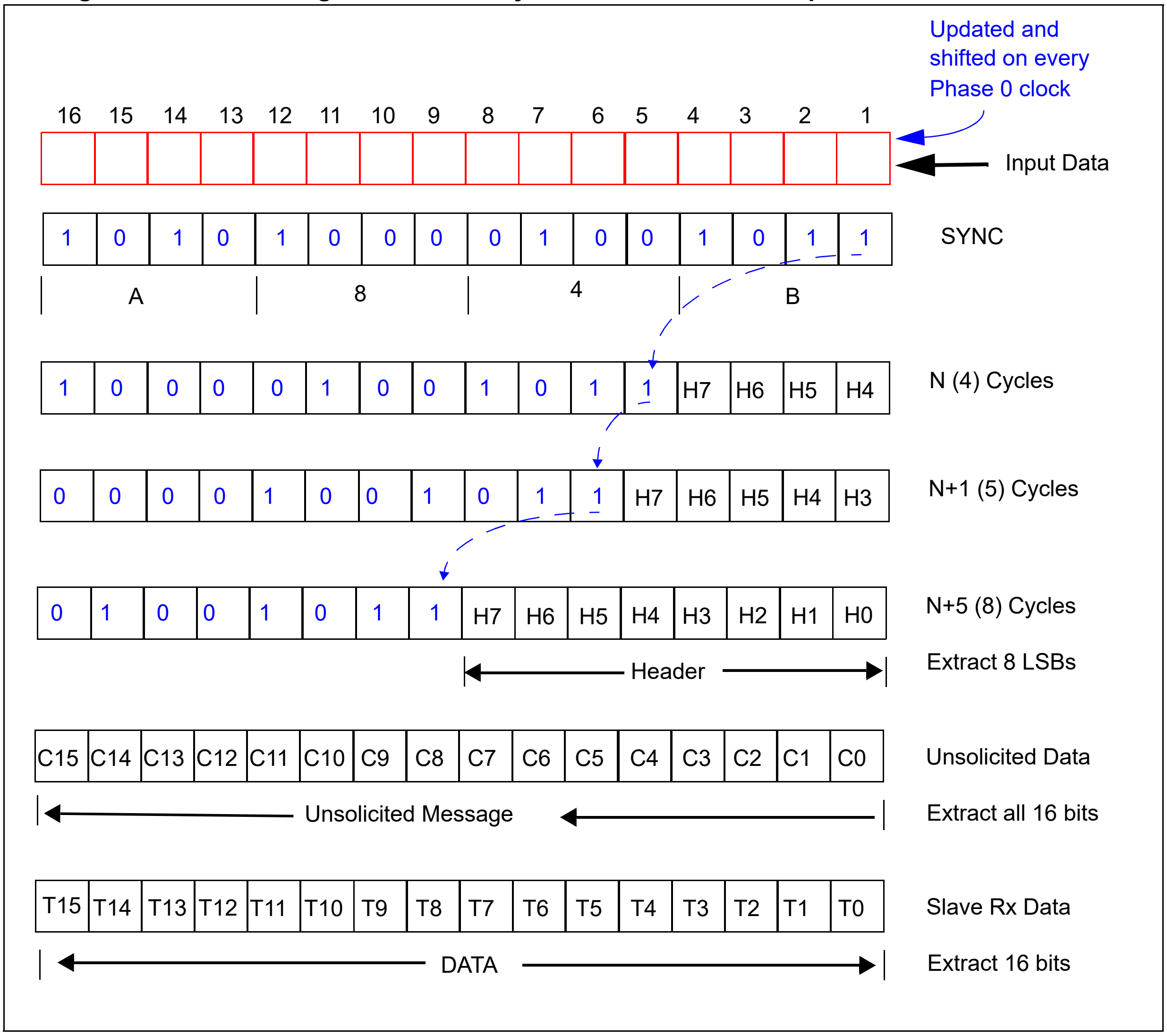
Figure 1363. Sampler 5 Logic



66.7.3.3 Header and Payload extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16-bit word, bit shifted every Phase 0 clock as shown in *Figure 1364*.

Figure 1364. Extracting Header and Payload from the 16-bit output from Auto-correlation



66.7.3.3.1 Line Receiver states

The LD has the following states:

- Shutdown
  The LR enters in the shutdown state when the power down signal is high. In this state the LR regulator is supplying a Vdd level, however the LR is not enabled. In this state the LR holds its off value. Once power down signal is negated, a predefined settling time is required before the LR may be used for communication. During the settling time the peer LFAST device should not start its transfer.
- Sleep
  The LR enters in the Sleep state when the sleep signal is high. In the sleep state, the LR is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In this

mode the receiver current consumption is reduced. Hysteresis in the LR ensures that it continues to present a 1 to the LFAST module.

- Normal
  In the Normal state, the LR is primed for reception. Receiver converts differential signals to a single ended logic signal.

Table 1341. Line Receiver state

LR State	LFAST Slave Triggers (at least one of the following)
Shutdown	– LFAST interface enable assertion/negation – Programing of LCR[SWOFFLR] or LCR{SWONLR] – Programing MCR[RXEN] and MCR[TXEN]
Sleep	– Programing of LCR[SWSLPLR] or LSC[SWWKLR]
Normal	Absence of above mentioned conditions

66.7.4 Transmit controller

66.7.4.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, such as unsolicited, ICLC, ping. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

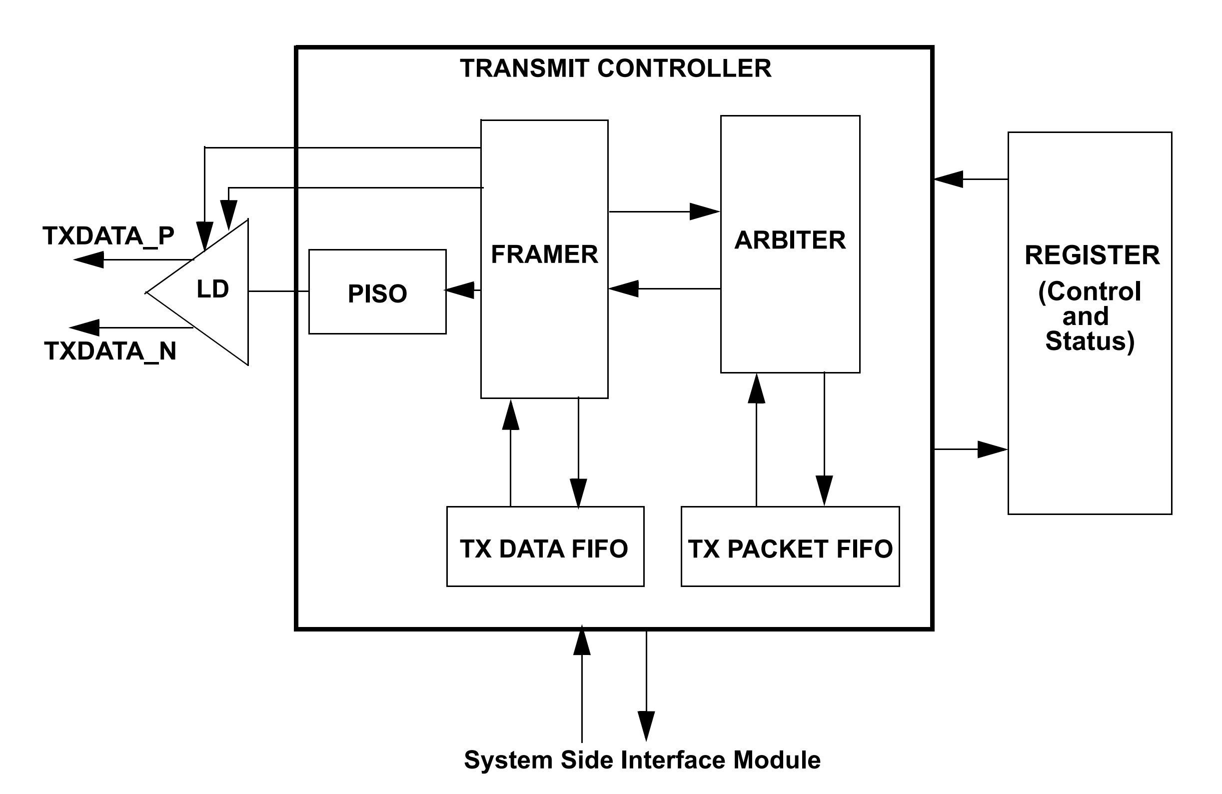
- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in *Figure 1365*.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

Figure 1365. Transmit controller connections



66.7.4.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like:

- Software programmable registers
- System side interface FIFO
- Rx interface controller
- LFAST Slave: LFAST interface enable (lfast_sysclk_en)

The level of priority for each request is shown in *Table 1342*.

Table 1342. Priority levels for the Transmit controller

Request	LFAST slave priority	Triggering conditions (at least one of the listed)
LFAST interface enable (lfast_sysclk_en)	1	– LFAST interface enable (lfast_sysclk_en) is asserted – LFAST interface enable (lfast_sysclk_en) is negated
Tx Interface disabled	2	– MCR[DRFEN] = 0 – MCR[TXEN] = 0 – MCR[DRFRST] = 1 – MCR[TXARBD] = 1 – LFAST Slave: ICLC frame with payload for “Disable Rx interface” received

Table 1342. Priority levels for the Transmit controller (continued)

Request	LFAST slave priority	Triggering conditions (at least one of the listed)
Tx Interface speed mode change	3	– SCR[TDR] is modified – LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	– TMCR[LPON] = 1 – LFAST Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
Ping response request	5	– LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 – LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	– Valid only if Last Frame with header b0 = 1 received from the peer LFAST – UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	– Tx FIFO contains one or more frames – Last Frame with header b0 = 1 received from the peer LFAST device – MCR[DATAEN] = 1
Clock mode test	8	– TMCR[CLKTST] = 0 – LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

Table 1343. Line driver states

LD state	LFAST slave triggers
Shutdown	– LFAST interface enable (lfast_sysclk_en) negation/assertion – Programing LVDS[SWOFFLD] and LVDS[SWONLD] – ICLC command from LFAST master – Programing MCR[TXEN] and MCR[DRFEN]
Sleep	– Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions

66.7.5 Frames supported

Table 1344 provides the frames supported and their permissible payload sizes

Table 1344. Frames supported by LFAST interfaces

Frame type	LCT code	Supported by 1. LFAST Master Tx 2. LFAST Slave Rx	Supported by 1. LFAST Slave Tx 2. LFAST Master Rx	Payload size (bits)
Data frame	Rx – 0100b,1000b – 1011b Tx – All	YES	YES	TX: 96 RX:128
Unsolicited frame	0001b	YES	YES	8, 32, 64, 96, 128, 256, 288
ICLC frame	0000b	YES	NO(1)	8
CTS frame	n/a	NO	NO	n/a
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

66.7.6 Frame flow

The following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

66.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

66.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

66.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer.

66.7.6.1.2 Data receive

LFAST master and slave support reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded.

66.7.6.2 Unsolicited flow

66.7.6.2.1 Unsolicited frame transmit flow

The software is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programmed into the Unsolicited Data and Control registers. Once the Payload is programmed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

66.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the *Mode Configuration Register (MCR)* to enable the Tx path.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
   – If UNSTCR[USNDRQ] = 1 then wait for either of the following:
     – UNSTCR[USNDRQ] = 0
     – TISR[TXUNSF] = 1.
   – If UNSTCR[USNDRQ] = 0 then:
     – Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
     – Program the unsolicited frame header in UNSTCR[UNSHDR].
     – Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.

66.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.

66.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is defined as a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

66.7.6.3.1 .ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

66.7.6.3.1.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF] = 1. The software can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame.The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

66.7.6.3.1.2 Start PLL ICLC

The Start PLL ICLC command will write bit RIISR[ICPONF] = 1.

66.7.6.3.1.3 Stop PLL ICLC

The Stop PLL ICLC command will write RIISR[ICPOFF] = 1.

66.7.6.3.1.4 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

66.7.6.3.1.5 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

66.7.6.3.1.6 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or software programs MCR[TXEN] = 1.

66.7.6.3.1.7 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

66.7.6.3.1.8 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

66.7.6.3.1.9 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable (lfast_sysclk_en).

66.7.7 Test and debug support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

66.7.7.1 Loopback test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for software should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programmed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 1345. Loopback modes

LPMOD[2:0]	Mode selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by the software, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

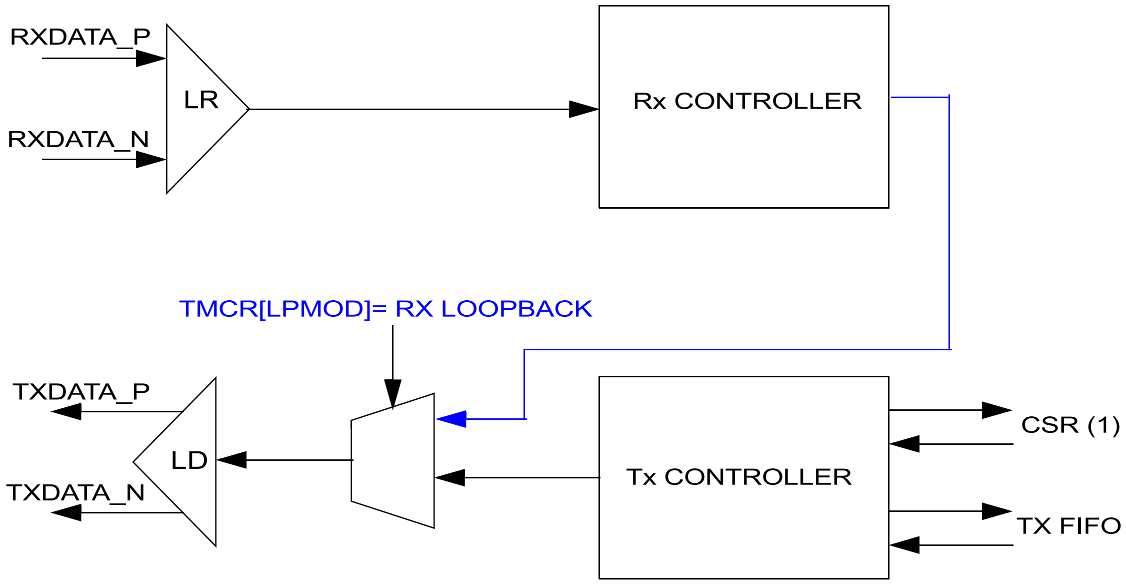
In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

66.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. Bit b0 of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

Figure 1366. Rx Loopback mode



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Rx Loopback mode:

1. Software programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - Software programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload 0xFF (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

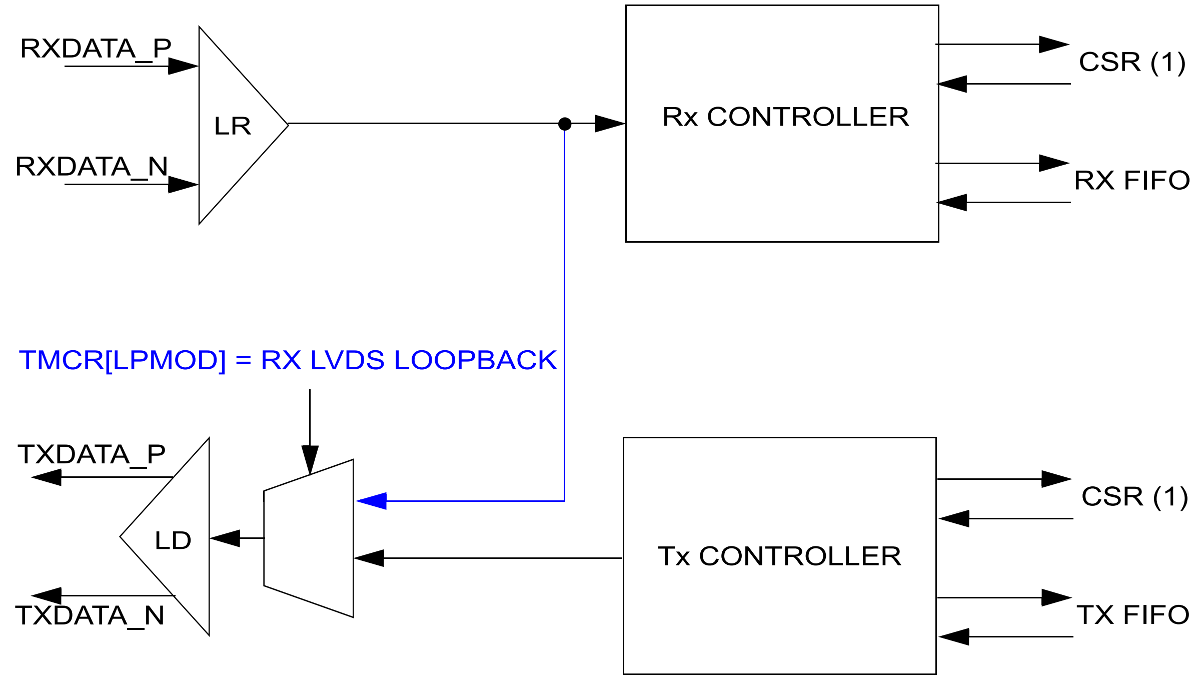
- Software programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 0x38 (Test mode off), from LFAST Master
- For LFAST Slave: - Deassertion of the LFAST interface enable (lfast_sysclk_en).

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

66.7.7.1.2 Rx LVDS Loopback mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via “Rx LVDS LoopBack Mux”. Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

Figure 1367. Rx LVDS LoopBack mode



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Rx LVDS Loopback mode:

1. Software programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:
   – The software writes TMCR[LPON] = 1.
   – For Slave Only: - Reception of ICLC Frame with Payload = 0xFF (Loopback mode ON), from LFAST Master.

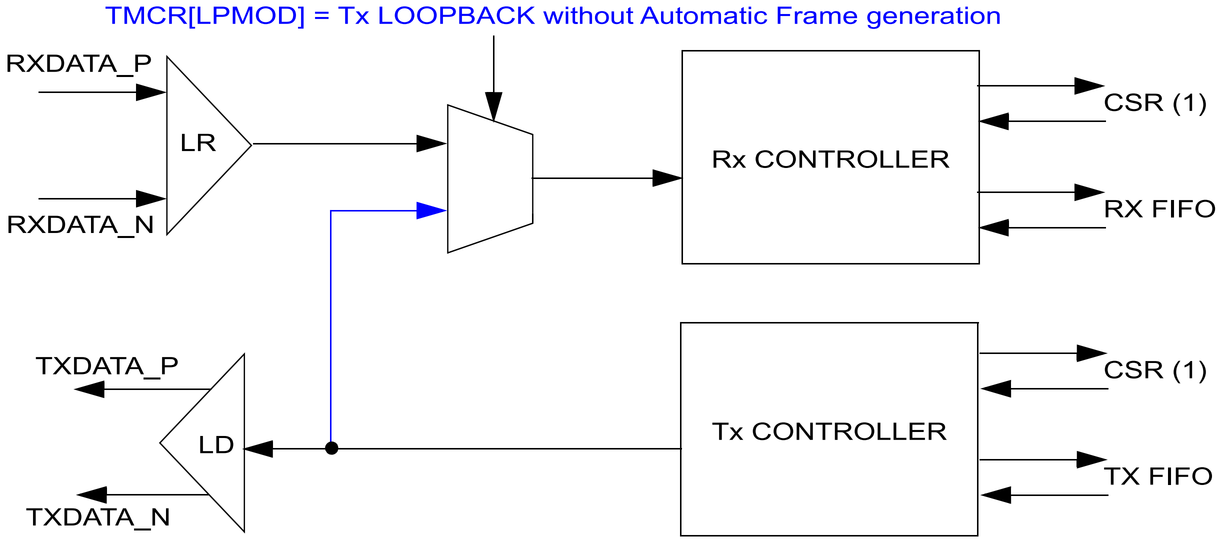
Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave:
  – Transmission of ICLC frame with payload 0x38 (Test mode off), from LFAST Master.
  – Deassertion of the LFAST interface enable (lfast_sysclk_en).

66.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

Figure 1368. Tx LoopBack mode without automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx Loopback without Automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   – Software programs TMCR[LPON] = 1.
   – For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 0x38 (Test mode off), from Tx interface (using unsolicited frame).
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

All frames and error flags are decoded in this mode.

66.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where predefined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal software intervention.

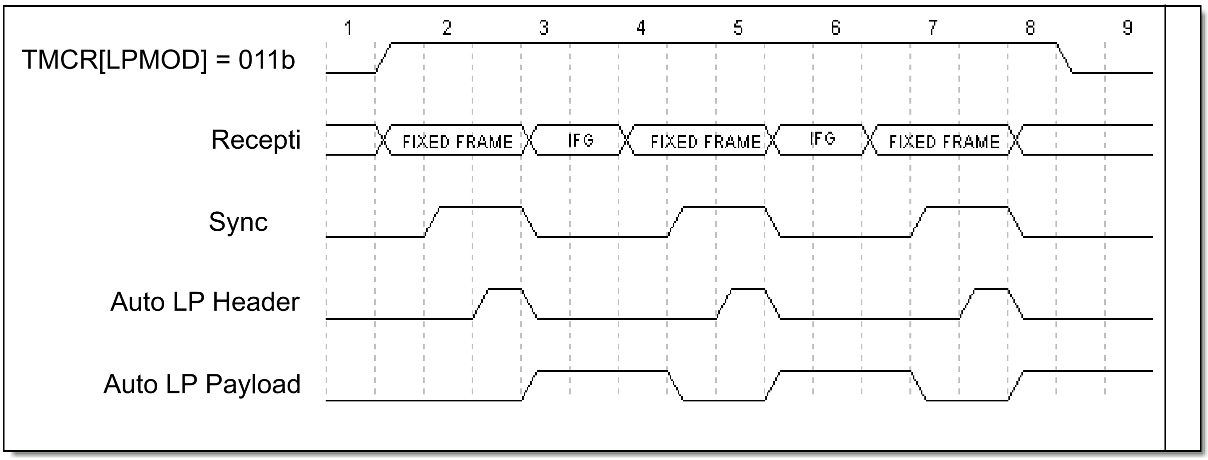
The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted.
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 0x13 received.
- GSR[LPCPDV]: Frame with Payload of 0xCB received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

Figure 1369. Automatic Loopback test status signal timings



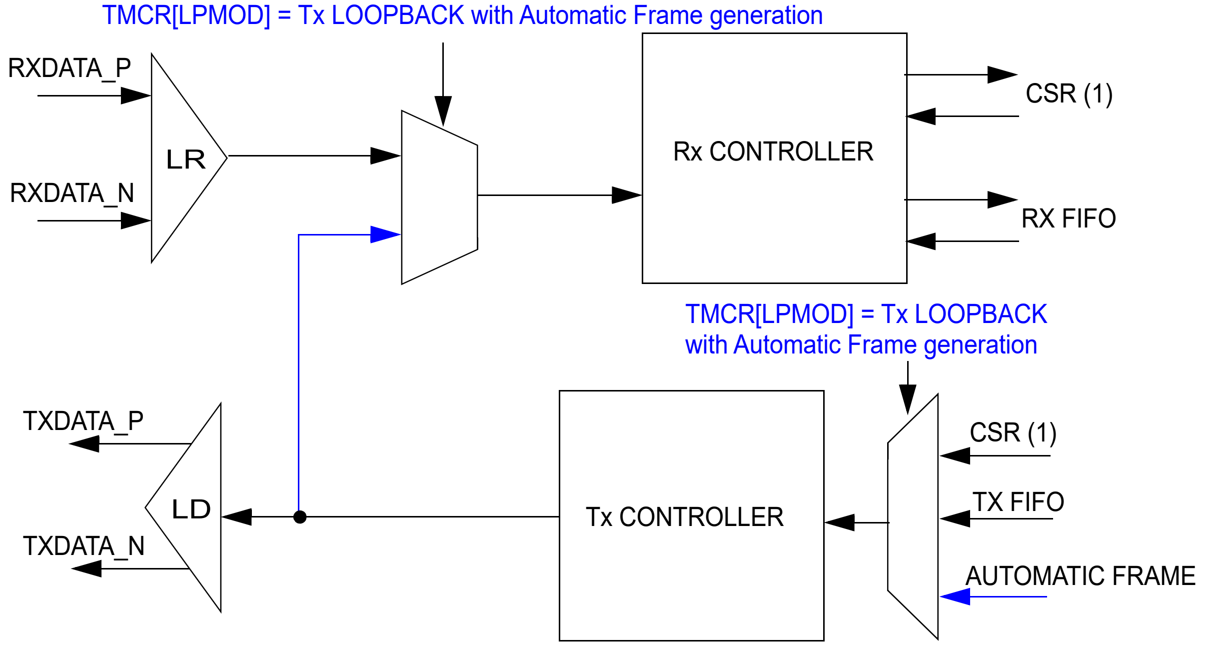
The two Loopback with automatic frame generation modes are:

- Tx Loopback with Automatic frame generation.
- Tx LVDS (external) with Automatic frame generation.

66.7.7.1.4.1 Tx LoopBack mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the software. The frames generated have fixed header of 0x13 and payload of 0xCB. The successful reception of this frame is indicated by the status signals and registers.

Figure 1370. Tx LoopBack mode with automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx Loopback with automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   - Software programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on), from the LFAST Master.

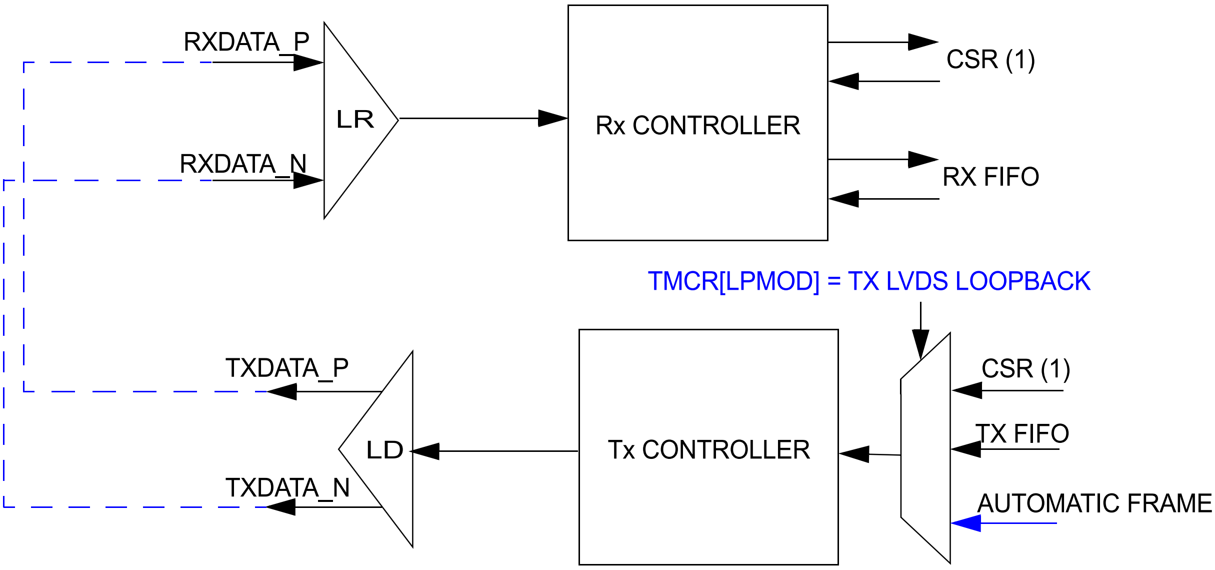
Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

66.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

Figure 1371. Tx LVDS Loopback (external) mode with automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx LVDS loopback with automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   – Software programs TMCR[LPON] = 1.
   – For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on) from LFAST Master

Exit from Tx LVDS Loopback with automatic frame generation mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

66.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- Software programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 0x34 (Clock Test mode on) from LFAST Master.

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be canceled by any of the following methods:

- Software programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 0x38 (Test mode off) from LFAST Master.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode does not affect the functionality of the Rx Interface.

66.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 1346. Frames supported by LFAST interfaces

Frame type	Tx buffer (in bits)	Rx buffer (in bits)	Memory type
Data frame	6 × 32 max 2 packets	8 × 32 max 2 packets	FIFO
Unsolicited frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC frame	N/A	1 × 8 max 1 packet(1)	Registers PICR[PNGPYLD]
CTS frame	N/A	N/A	N/A

1. Ping response packet.

66.9 Resets

The various blocks in LFAST are reset as described in *Table 1347*.

Table 1347. Recommended receive exception handling mechanism

Reset	Blocks reset
Asynchronous Hardware reset	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFRST bit of *Mode Configuration Register (MCR)*	Polarity: Active High – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space

Table 1347. Recommended receive exception handling mechanism (continued)

Reset	Blocks reset
DRFEN bit of *Mode Configuration Register (MCR)*	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ]
LFAST interface enable (lfast_sysclk_en)	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers except the following: TISR, RISR, RIISR, GSR[LPTXDN] and GSR[LPFPDV] These bits remain reset while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: Negedge only – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ] These bits do not remain reset while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: positive edge only – TISR, – RISR, – RIISR, – GSR[LPTXDN] and GSR[LPFPDV] These bits do not remain reset while the LFAST interface enable (lfast_sysclk_en) is High.

66.10 Clocks

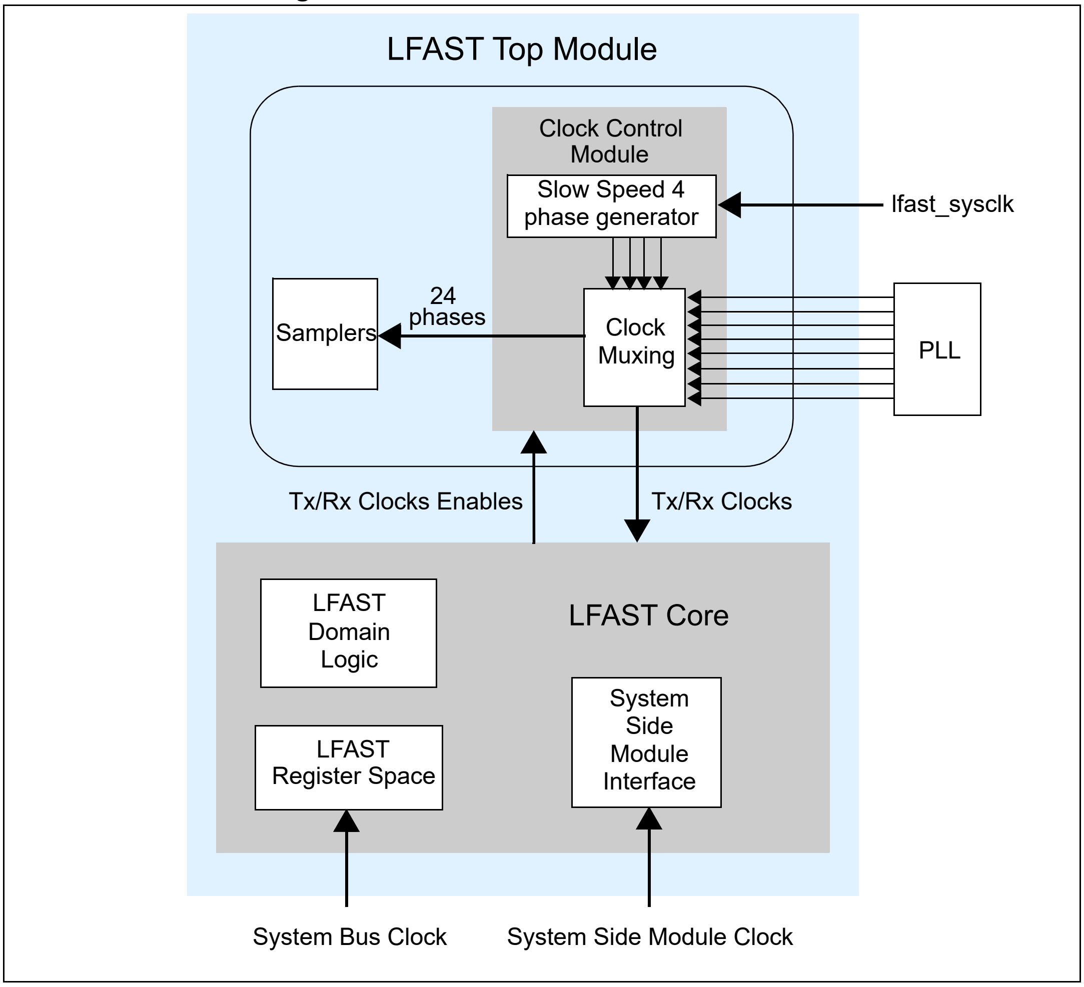
The LFAST mainly works on three clocks:

- System Bus Clock, used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

66.10.1 Clocking strategy

Figure 1372 shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

Figure 1372. LFAST module clock domains

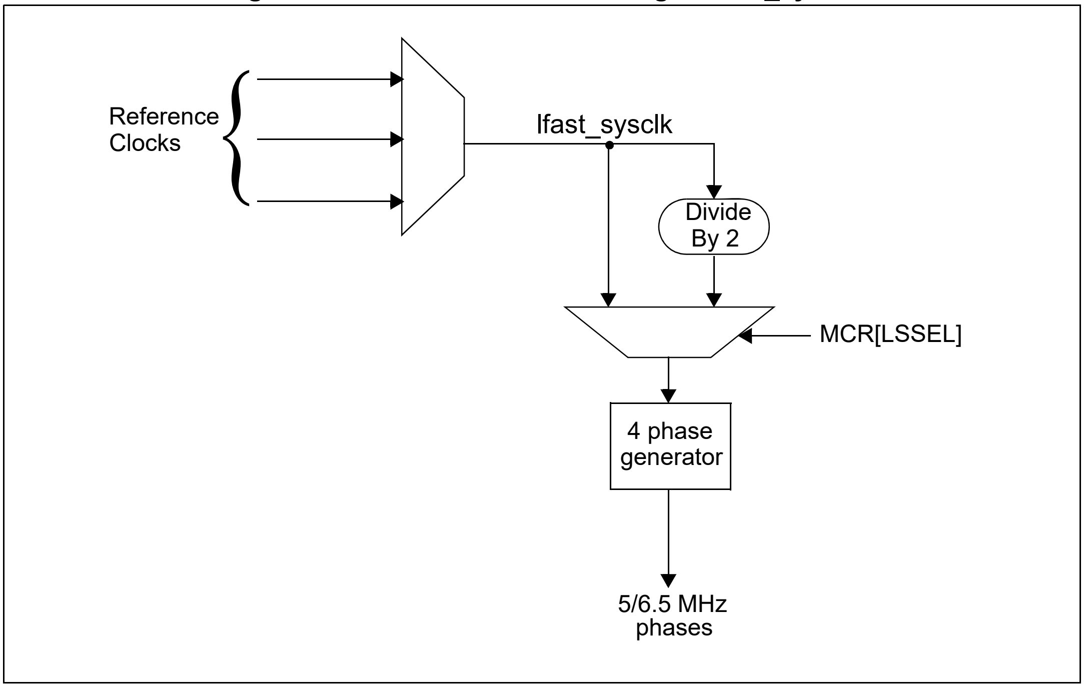


66.10.2 Slow speed clock

66.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in *Figure 1373*, *Figure 1374* and *Figure 1375*.

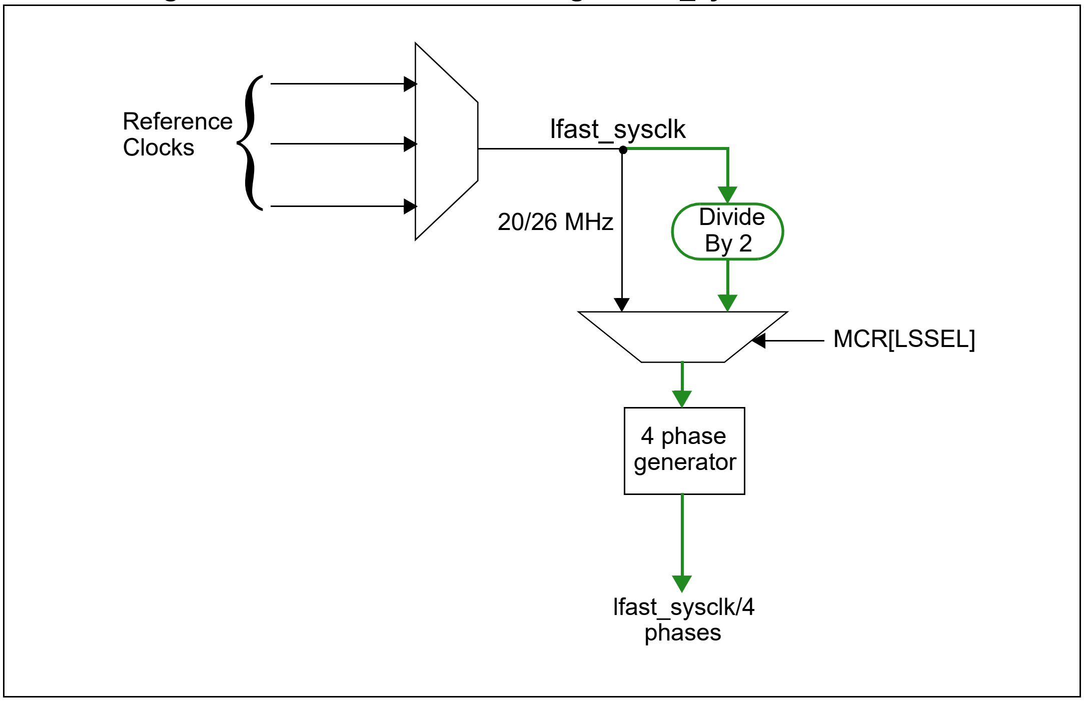
Figure 1373. External clock muxing of lfast_sysclk



All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in *Figure 1373*.

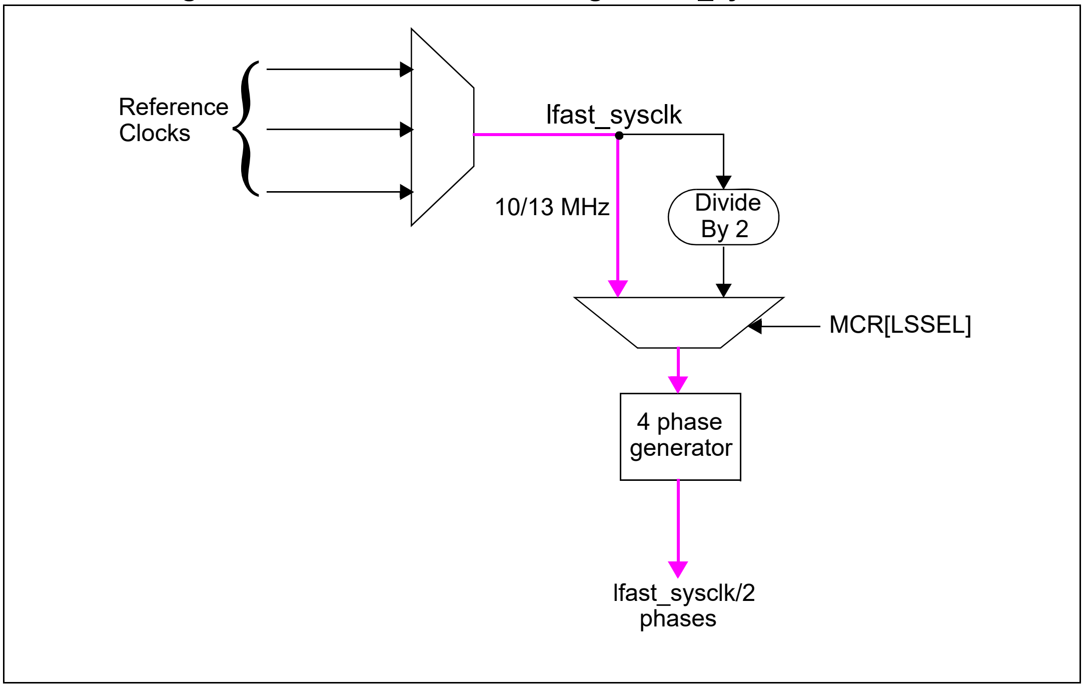
The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in *Figure 1374* (selected clock path shown in green).

Figure 1374. External clock muxing of lfast_sysclk of 20/26 MHz



In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in *Figure 1375* (selected clock path selected shown in magenta).

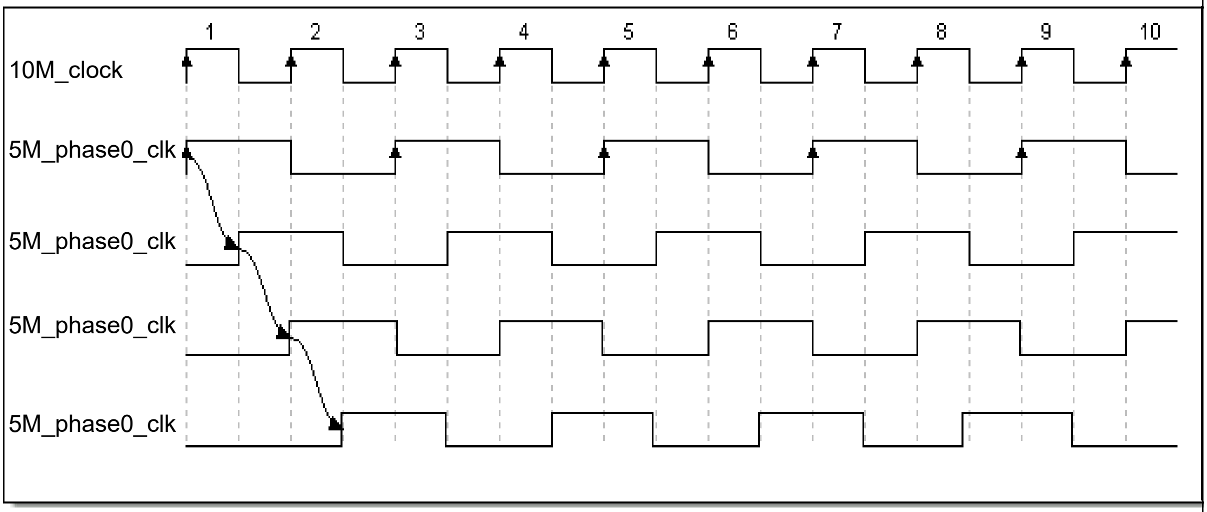
Figure 1375. External clock muxing of lfast_sysclk of 10 MHz



66.10.2.2 Slow speed 4-phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. *Figure 1376* Shows 4 phases getting generated using lfast_sysclk.

Figure 1376. Slow speed 4 phases generation

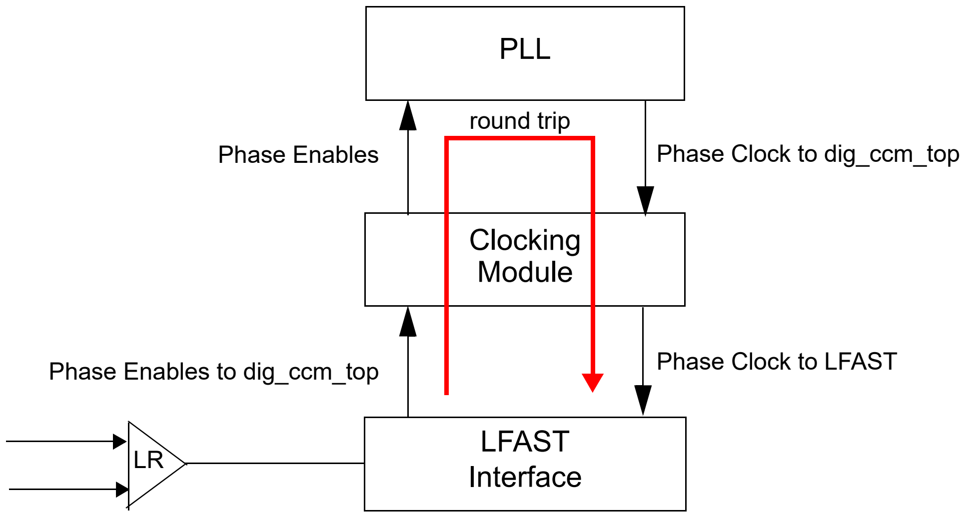


66.10.3 Rx controller clocks

66.10.3.1 Clocking module requirements for high speed phases

The high speed Phases are obtained from the PLL via the Clocking Module as shown in *Figure 1377*.

Figure 1377. Clock enables and clock paths



The LFAST block will know which of the phases will be required for the different modes of operations. Therefore, the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

66.10.3.2 Clock module requirements for low speed phases

66.10.3.2.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 1348. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

66.10.4 Tx controller clocks

66.10.4.1 Clocking module requirements for speed phases

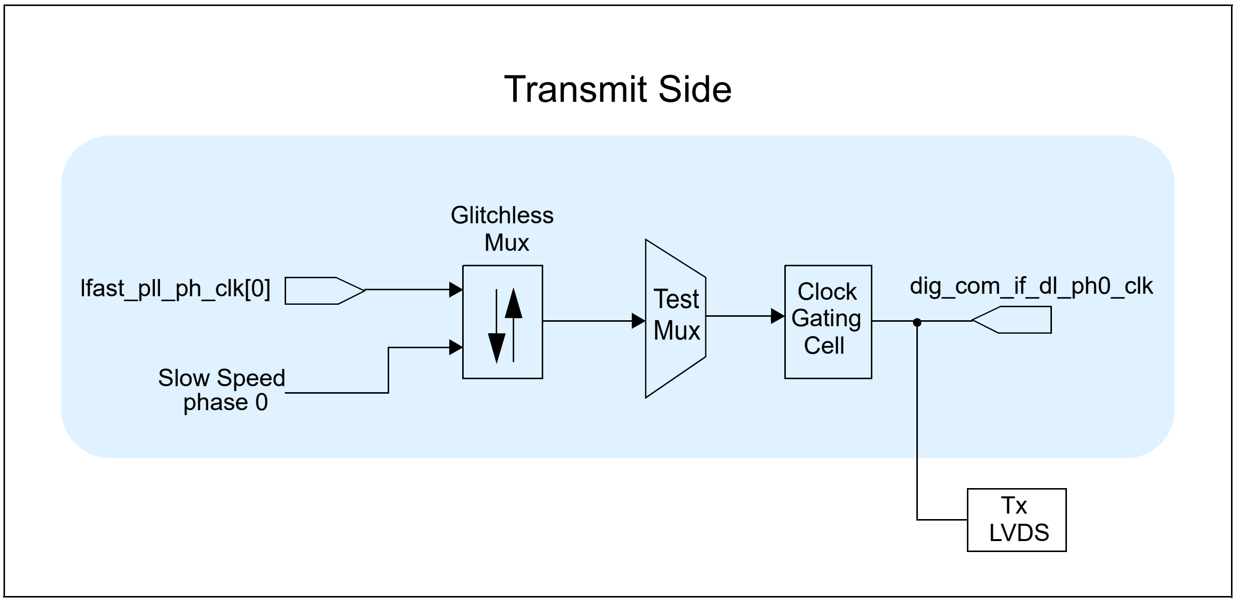
The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See *Table 1349*.

Table 1349. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

66.10.4.2 Transmit clock muxing

Figure 1378. Transmit clocks muxing



66.10.4.3 Tx request clock control

The Tx phase 0 clock is enabled when all the resets are negated.

67 Development Trigger Semaphore (DTS)

67.1 Introduction

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note: When used as a component of a triggered data acquisition system, Nexus read/write access is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus Auxiliary port.

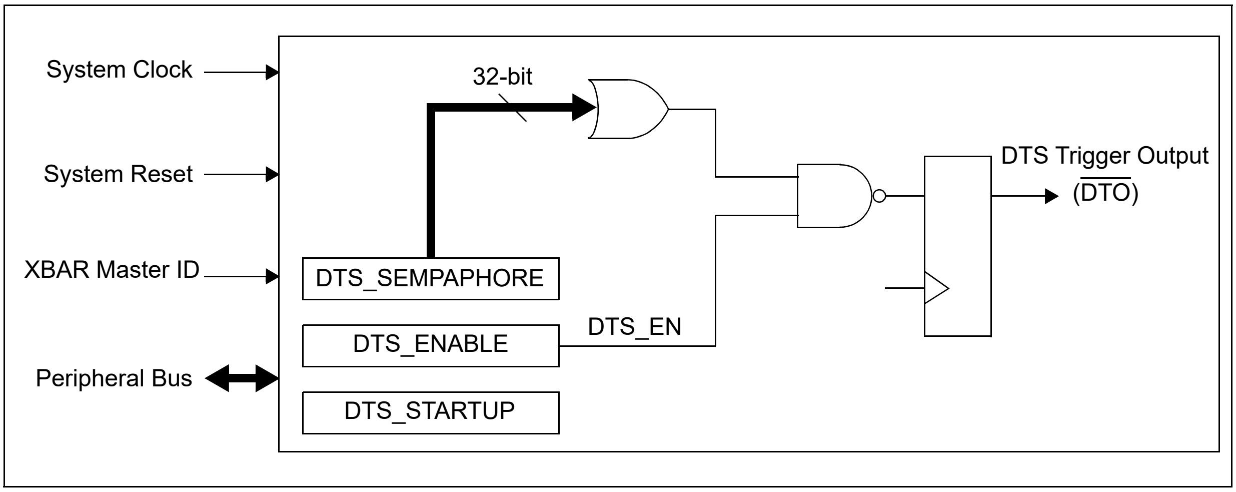
67.2 Overview

The Development Trigger Semaphore (DTS) module consists of registers and a small amount of combinational logic to generate an output signal, DTS Trigger Output ($\overline{\text{DTO}}$). The registers are as follows:

- DTS_SEMAPHORE register: Any bit in this 32-bit register, when set to a value of logic ‘1’, causes the DTS module output signal to be asserted if the corresponding DTS_EN bit is set in the DTS_ENABLE register. This enables an external tool to detect up to 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access or debug Zipwire.
- DTS_STARTUP register: This register provides a mechanism for the external tool to notify software running on the CPU that the tool is connected and can provide information about either the type of tool or options that can be used by the software.
- DTS_ENABLE register: This register provides an enable/disable capability for the DTS feature.

The architecture is shown in *Figure 1379*.

Figure 1379. DTS block diagram

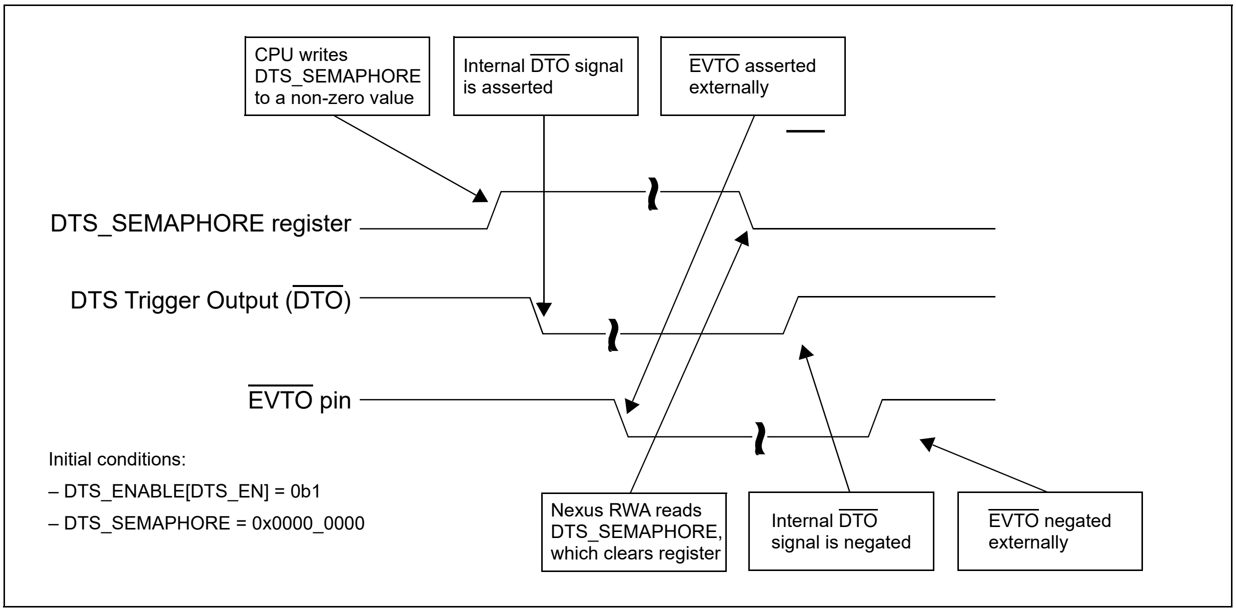


The DTS Trigger Output (DTO) signal is connected to one of the EVTO inputs of the Debug and Calibration Interface (DCI). The other EVTO inputs are connected to the other Nexus modules in the device. DTO is asserted when any bit in a semaphore register is set and the DTS function of that register is enabled.

Note: When the DTS module is enabled (DTS_ENABLE[DTS_EN] = 0b1), all other Nexus EVTO functions associated should be disabled by the tool and EVTO becomes the DTO. Unlike the EVTO function that only asserts for one clock, the DTO function remains asserted until the tool reads the semaphore register that is the source of the assertion, clearing the register's contents.

Figure 1380 shows an example of the chain of events that begins with setting of a bit in the DTS_SEMAPHORE register and the clearing of the register caused by a Nexus read.

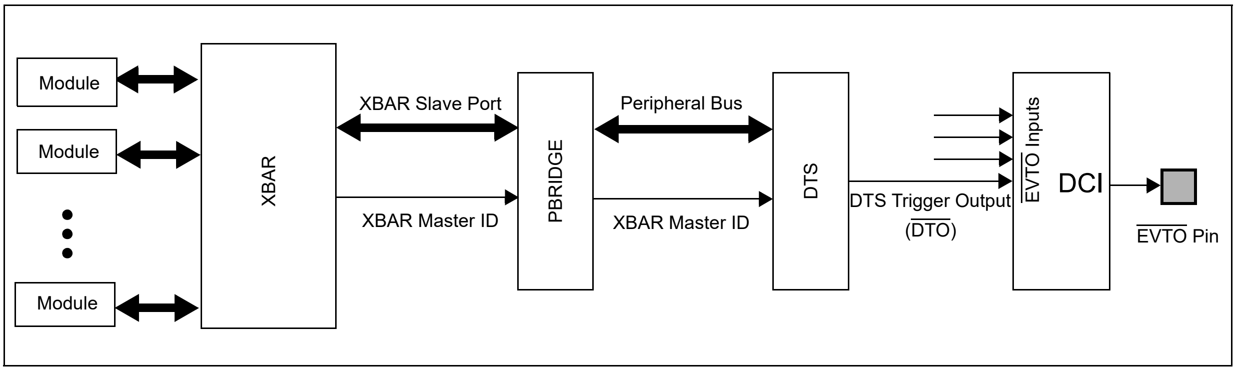
Figure 1380. DTO event sequence example



67.3 DTS device connections

Figure 1381 shows the DTS device connections. The DTS module connects to the Peripheral Bridge (PBRIDGE) for access to the registers. The PBRIDGE is connected to the device modules through a slave port of the Crossbar bus interface (XBAR).

Figure 1381. DTS device connections



The registers have limited access as described in *Section 67.3.1: DTS register access*. Access is based on the XBAR Master ID of the accessing module. Writing to the semaphore register is limited to the cores and the eDMA module and is restricted to only setting bits. Only an access via a Nexus Read/Write Access from an external tool or debug Zipwire can clear bits in the semaphore register (semaphore register bits are cleared automatically when read via Nexus/Write Access). Similarly, the DTS_ENABLE and DTS_STARTUP registers can only be written via a Nexus Read/Write Access or debug Zipwire.

Note: Nexus Read/Write Accesses use the core load/store bus to perform accesses, but Nexus accesses have a different Master ID than normal core load/stores.

67.3.1 DTS register access

A summary of accesses to all DTS registers by bus masters is provided in *Table 1350*. Only proper 32-bit accesses are valid. The effects of write accesses that are not 32 bits are not defined.

Table 1350. DTS register access effects

Register	32-bit read				32-bit write			
	RWA(1)	Core	eDMA	Other crossbar masters	RWA(1)	Core	eDMA	Other crossbar masters
DTS_ENABLE	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_STARTUP	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_SEMAPHORE	Data and Clear(2)	Data	Data	Data	No effect	Bit OR	Bit OR	No effect

1. Nexus Read/Write access via an external tool
2. A read of the semaphore register by Nexus Read/Write Access module is destructive and clears all bits in the register

Access to DTS module registers is controlled based on the XBAR Master ID of the accessing module.

Note: The XBAR Master ID should not be confused with the Master Port number of the XBAR. See Chapter 17: Crossbar switch (XBAR), for details.

Tools must access the DTS registers through the Nexus Read/Write Access mechanism of a core or debug Zipwire. External tool accesses through either core appear as if the access is via the core and therefore do not have the same level of access as a Nexus Read/Write Access.

67.4 Memory map and register definition

Table 1351 shows the memory map of the DTS module registers.

Table 1351. DTS memory map

Offset address	Register name	Access	Reset Value	Location
0x0000	DTS Output Enable register (DTS_ENABLE)	Restricted R/W(1)	0x0000_0000	Section 67.4.1.1
0x0004	DTS Startup register (DTS_STARTUP)	Restricted R/W(1)	0x0000_0000	Section 67.4.1.2
0x0008	DTS Semaphore Register (DTS_SEMAPHORE)	Restricted R/W(1)	0xFFFF_FFFF	Section 67.4.1.3
0x000C	DTS Semaphore Extension (DTS_SEMAPHORE_B)	Restricted R/W(1)	0xFFFF_FFFF	Section 67.4.1.4

1. Only certain types of accesses are allowed. See the register figure note of each register for a description of the accesses that are allowed.

67.4.1 Register descriptions

67.4.1.1 DTS output enable register (DTS_ENABLE)

The DTS_ENABLE register controls the DTS Trigger Output ($\overline{\text{DTO}}$) and whether $\overline{\text{DTO}}$ is active on the $\overline{\text{EVTO}}$ output pin of the device. Figure 1382 shows the format of the DTS_ENABLE register. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA) or debug Zipwire.

Note: Access to the DTS_SEMAPHORE and DTS_STARTUP registers are unaffected by the state of this register.

Offset 0x0000 Access: Restricted R/W(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DTS_EN_B	DTS_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA).

Figure 1382. DTS output enable register (DTS_ENABLE)

Table 1352. DTS_ENABLE field descriptions

Field	Description
30 DTS_EN_B	DTS Enable B Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE_B register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE_B register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).
31 DTS_EN	DTS Enable Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).

67.4.1.2 DTS startup register (DTS_STARTUP)

Figure 1383 shows the format of the DTS_STARTUP register, used for tool detection and startup information exchange between external data acquisition tool and the embedded controller. DTS_STARTUP register can be read by the core, the eDMA modules and other crossbar masters but can only be updated by a Nexus Read Write Access (RWA) or debug Zipwire.

Offset 0x0004 Access: Restricted R/W(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	AD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The DTS_STARTUP register can be read by the cores, the eDMA module and Nexusbut can only be updated by a Nexus Read Write Access (RWA).

Figure 1383. DTS startup register (DTS_STARTUP)

Table 1353. DTS_STARTUP field descriptions

Field	Description
0:31 AD	Application Dependent register bits. The bits have no defined meaning to the microcontroller. They are used to by an external tool to pass information (for example, application options and status) to application software running on target microcontroller at startup time. Use a Nexus RWA 32-bit write access or debug Zipwire to update the contents of this register. **Note:** A device reset (either from the $\overline{\text{RESET}}$ pin or an internally generated reset) clears all bits in the register. A Nexus reset does not change the contents of the register.

67.4.1.3 DTS semaphore register (DTS_SEMAPHORE)

Figure 1384 shows the format of the DTS_SEMAPHORE register. The DTS_SEMAPHORE register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Offset 0x08 Access: Restricted R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ST															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1384. DTS semaphore register (DTS_SEMAPHORE)

Table 1354. DTS_SEMAPHORE field descriptions

Field	Description
0:31 ST	Semaphore Trigger. When a core or eDMA writes a logical ‘1’ to a bit, the bit is set. A write of ‘0’ by the core or DMA does not change the state of the bit. – All register bits are set to ‘1’ by a device reset. – A Nexus reset does not change the state of this register. – The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA or debug Zipwire. – For the core or DMA, only 32-bit write or read accesses are valid. – A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

67.4.1.4 DTS semaphore extension (DTS_SEMAPHORE_B)

Figure 1385 shows the format of the DTS_SEMAPHORE_B register. The DTS_SEMAPHORE_B register is used by software to assert the DTO signal on the device EVTO pin. A one in any bit of this register causes the DTO signal on the EVTO pin to be driven low. The intended use of this register is for the DTO signal to notify tools that data is available. Individual bits are used to identify the specific data.

Offset 0x0C Access: Restricted R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ST_B															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST_B															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1385. DTS Semaphore extension register (DTS_SEMAPHORE_B)

Table 1355. DTS_SEMAPHORE_B field descriptions

Field	Description
0:31 ST_B	Semaphore Trigger Extension. When a core or eDMA writes a logical ‘1’ to a bit, the bit is set. A write of ‘0’ by the core or DMA does not change the state of the bit. – All register bits are set to ‘1’ by a device reset. – A Nexus reset does not change the state of this register. – The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA or debug Zipwire. – For the core or DMA, only 32-bit write or read accesses are valid. – A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

67.5 Example application

The calibration process of a new engine requires real-time access to calibration tables and the ability to update the tables in real-time The DTS module enables this capability by enabling software to assert a signal to an external device pin to notify an external tool that data is available. The tool can then retrieve the data.

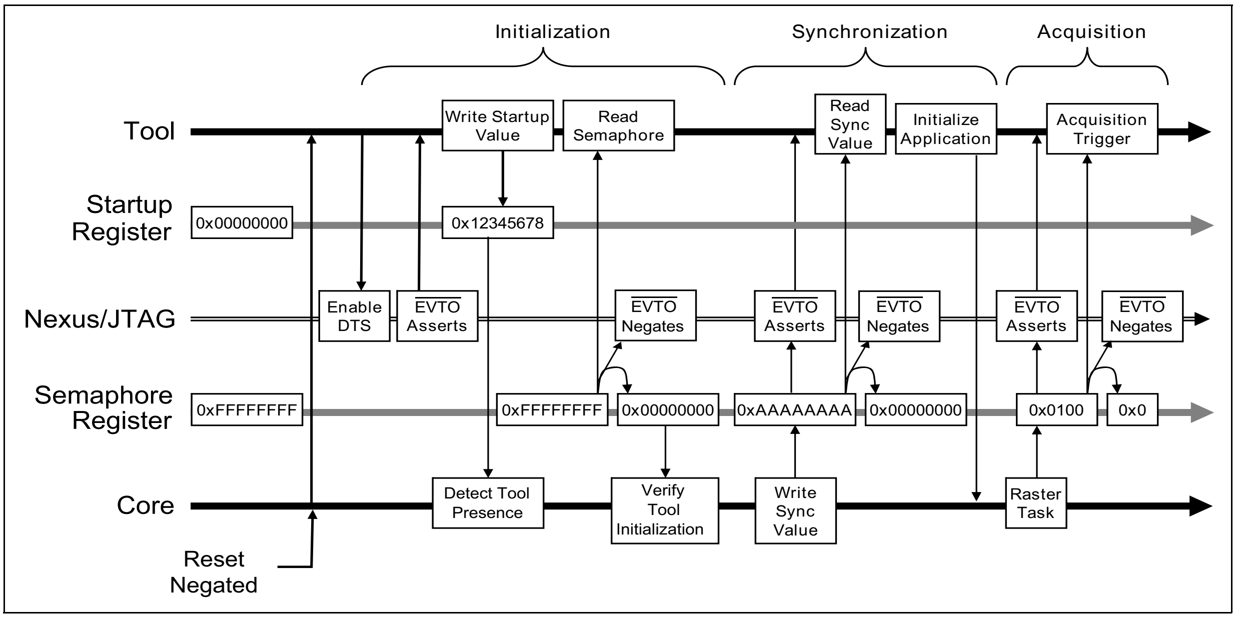
In this type of application the DTS_SEMAPHORE register and DTS Trigger Output (DTO) signal provide a mechanism to notify the calibration tool that the calibration variable or variables (or sets of measurements), up to 32, have been updated with new values and are available for the tool to access.

Note: It is the user’s responsibility to ensure that the tool has time to retrieve the data prior to that particular trigger being set a second time. It is also permissible to have multiple triggers *active at the same time or for a second trigger to be set before a previous trigger has been serviced, as long as it is not the same trigger (unless it is acceptable to the tool not to receive every data set).*

Figure 1386 shows an example of DTS startup sequence for an external real-time data acquisition system. The startup and synchronization sequence can be as simple or as complicated as the need requires. However, a typical startup sequence is as follows:

1. The DTS_STARTUP register is cleared by a power on reset or any CPU reset.
2. The tool writes a non-zero value to the DTS_STARTUP register.
3. The CPU (user application software) then reads the value of the DTS_STARTUP register. Based on this value, different initialization options can be selected. The bits can be used for any application specific definitions.
4. Since the DTS_SEMAPHORE register is cleared when the tool reads the current value. The tool should perform all necessary initialization before reading this register. The application software can then check that the DTS_SEMAPHORE register was cleared by the tool, to determine that it is safe to start using it for its intended raster trigger semaphore function.
5. An optional hand shake from the CPU can be used to inform the tool that the user software has detected that the tool is attached and the CPU has performed the proper initialization for the tool by writing a predefined value to the DTS_SEMAPHORE register (the example shown in the figure above uses 0xAAAA_AAAA, all A's was used since it is unrealistic that 16 channels could be enabled very quickly after start up after a reset).

Figure 1386. DTS startup sequence example



68 Nexus Port Controller (NPC)

68.1 Information specific to this device

This section presents device-specific parameterization, customization, and feature availability information not specifically referenced in the remainder of this chapter.

68.1.1 Parameter values

The following table provides the parameter values for this device.

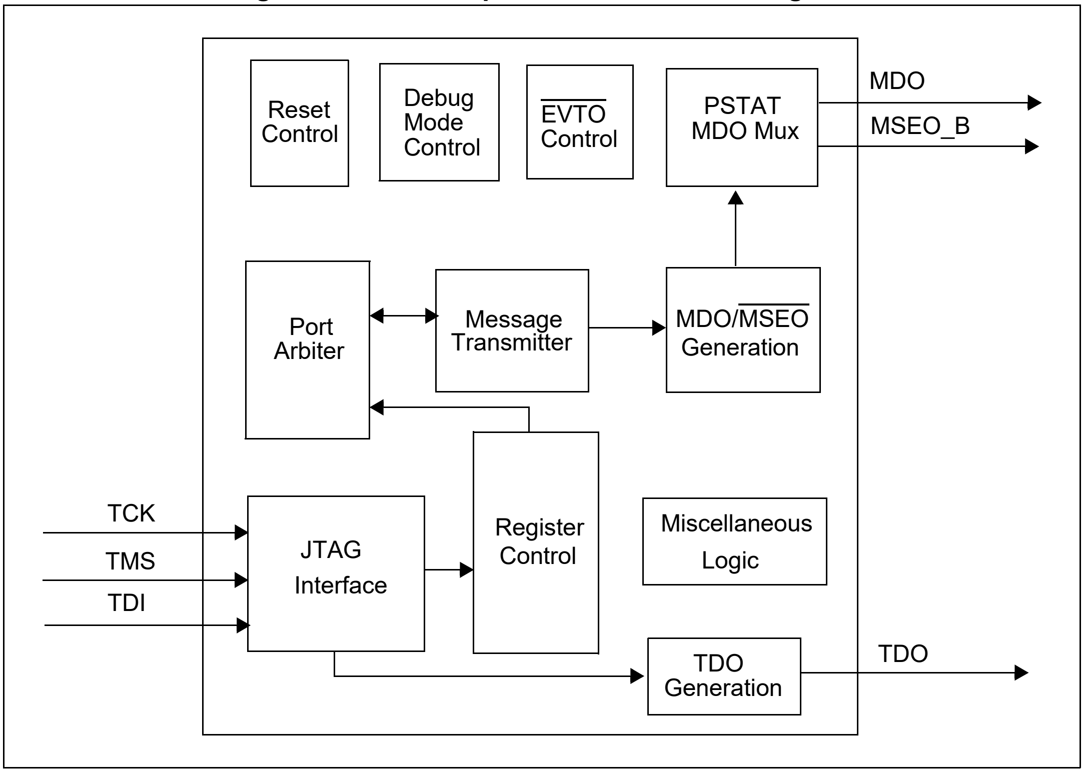
Table 1356. Device parameter values

Parameter	Value
Number of MDO pins available in reduced-port mode	4
Number of MDO pins available in full-port mode	12
Number of $\overline{\text{MSEO}}$ pins available	2

68.2 Introduction

The following figure shows the Nexus Port Controller (NPC) block diagram.

Figure 1387. Nexus port controller block diagram



68.2.1 Overview

On a system-on-a-chip device, there are often multiple blocks that require development support. Each of these blocks implements a development interface based on the IEEE-ISTO 5001-2003 standard. The blocks share input and output ports that interface with the development tool. The NPC controls the usage of the input and output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

68.2.2 Features

The NPC block performs the following functions:

- Controls arbitration for ownership of the Nexus Auxiliary Output Port
- Nexus Device Identification Register and Messaging
- Generates MCKO enable and frequency division control signals
- Controls sharing of $\overline{\text{EVTO}}$
- Generates an MCKO clock gating control signal to enable gating of MCKO when the auxiliary output port is idle
- Generates power-on reset status
- Provides Nexus support for censorship mode
- $\overline{\text{RDY}}$ pin support to increase the transfer rate of the IEEE 1149.1 port for large data read requests

68.2.3 Modes of operation

The NPC block uses the censorship status and an internal power-on reset indication as its primary reset signals. Upon exit of reset, the mode of operation is determined by the Port Configuration Register (PCR) settings.

68.2.3.1 Reset

The NPC block is asynchronously placed in reset when either power-on reset is asserted, the device enters censored mode, or the TAP controller state machine is in the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. The NPC is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state
- The auxiliary output port pins are negated
- The TDI, TMS, and TCK TAP inputs are ignored (when in power-on reset or censored mode only)
- Registers default back to their reset values

68.2.3.2 Disabled-Port mode

In disabled-port mode, auxiliary output pin port enable signals are negated, thereby disabling message transmission. Any debug feature that generates messages cannot be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through $\overline{\text{EVTO}}$.

68.2.3.3 Full-Port mode

Full-port mode (FPM) is entered by asserting the MCKO_EN and FPM bits in the PCR. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

68.2.3.4 Reduced-Port mode

Reduced-port mode (RPM) is entered by asserting the MCKO_EN bit and deasserting the FPM bit in the PCR. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

68.2.3.5 Censored mode

When the device is in censored mode, reading the contents of internal flash externally is not allowed. To prevent Nexus modules from violating censorship, the NPC is held in reset when in censored mode, asynchronously holding all other Nexus modules in reset as well. This prevents Nexus read/write to memory mapped resources and the transmission of Nexus trace messages.

68.2.3.6 Nexus Double Data Rate mode

Nexus double data rate (DDR) mode is enabled by asserting the DDR_EN bit in the PCR. DDR mode is only supported for MCKO_DIV settings of sys_clk/2, sys_clk/4 and sys_clk/8. In double data rate mode, message data is updated between the edges (both rising and falling) of MCKO, effectively doubling message throughput.

68.3 External signal description

68.3.1 Overview

The NPC pin interface provides for the transmission of messages from Nexus blocks to the external development tools and for access to Nexus client registers. The NPC pin definition is outlined in the following table.

Table 1357. NPC signal properties

Name	Port	Function	Reset state	Pull(1)
EVTO	Auxiliary	Event Out pin	0b1	—
MDO	Auxiliary	Message Data Out pins	0	—
MSEO	Auxiliary	Message Start/End Out pin	0b1	—
TCK	JTAG	Test Clock Input	—	Down
TDI	JTAG	Test Data Input	—	Up
TDO	JTAG	Test Data Output	High Z(2)	—
TMS	JTAG	Test Mode Select Input	—	Up
RDY	JTAG	Data ready for transfer to/from NRRs	—	—

1. The pull is not implemented in this block. Pullup/pulldown devices are implemented in the pads.
2. TDO output buffer enable is negated when the NPC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented on TDO at the SoC level.

68.3.2 Detailed signal descriptions

This section describes each of the signals listed in *Table 1357* in more detail. The JTAG test clock (TCK) input from the pin is not a direct input to the NPC. The NPC requires two separate input clocks for TCK clocked logic, one for posedge (rising edge TCK) logic and one for negedge (falling edge TCK) logic. Both clocks are derived from the pin TCK, and generated external to the NPC.

68.3.2.1 $\overline{\text{EVTO}}$ - Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus blocks that implement the signal.

68.3.2.2 MDO - Message Data Out

Message Data Out (MDO) are output pins used for uploading OTM, BTM, DTM, and other messages to the development tool. The development tool should sample MDO on the rising edge of MCKO. The width of the MDO bus used is determined by reset configuration.

68.3.2.3 MSEO_B - Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is one output pin that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool should sample $\overline{\text{MSEO}}$ on the rising edge of MCKO.

68.3.2.4 TCK - Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

68.3.2.5 TDI - Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

68.3.2.6 TDO - Nexus Test Data Output

Test Data Output (TDO) pin transmits serial output for instructions and data. TDO is three-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

68.3.2.7 TMS - Test Mode Select

Test Mode Select Input (TMS) pin is used to sequence the IEEE 1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

68.3.2.8 $\overline{\text{RDY}}$ - Data ready for transfer

The $\overline{\text{RDY}}$ pin exists to increase the transfer rate of the IEEE 1149.1 port. It is used to signal when data are ready to be transferred to and from NRRs. This may eliminate the need to poll NRRs for status information for synchronization purposes. This capability becomes especially important when performing read/write access transfers to different speed target memories.

The $\overline{RDY}$ pin asserts (asynchronously) a logic low whenever the read/write access transfer has completed without error and then deasserts when the IEEE 1149.1 state machine has reached the CAPTURE_DR state.

68.4 Register definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

Table 1358 shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC block does not implement the client select control register because the value does not matter when accessing the registers. Note that the bypass and instruction registers have no index values. These registers are not accessed in the same manner as Nexus client registers. Refer to the individual register descriptions for more detail.

Table 1358. NPC registers

Index	Register
0	Device ID Register (DID)
127	Port Configuration Register (PCR)

68.4.1 Register descriptions

This section contains the NPC register descriptions.

68.4.1.1 Bypass Register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

68.4.1.2 Instruction Register

The NPC block uses a 4-bit instruction register as shown in *Figure 1388*. The instruction register is accessed via the SELECT_IR_SCAN path of the tap controller state machine, and allows instructions to be loaded into the block to enable the NPC for register access (NEXUS_ENABLE) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in synchronous loading of the BYPASS instruction. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register’s read value when the TAP controller is sequenced into the Shift-IR state.

Figure 1388. 4-bit Instruction Register

	0	1	2	3
R	Previous Instruction Opcode			
W	Instruction Opcode			
Reset:	BYPASS Instruction Opcode (0xF)			

68.4.1.3 Nexus Device ID Register (DID)

The device identification register allows identification information of the part to be determined through the auxiliary output port.

Register index: 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	Manufacturer_ID											0	Sub_Family		
W																
Reset	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Sub_ Family	Application_Family			Technology _Node		Design_Center						Minor_Mask			
W																
Reset	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 1389. Nexus Device ID Register

Table 1359. DID field descriptions

Field	Description
1:11 Manufacturer_ID	These bits contain the Joint Electron Device Engineering Council (JEDEC) manufacturer's identification code. 00000100000 STMicroelectronics
13:16 Sub_Family	0001 SPC58xCx
17:19 Application_Family	010 SPC58x - Chassis & Safety
20:21 Technology_Node	01 40 nm
22:27 Design_Center	These bits indicate the device Design Center.
28:31 Minor_Mask	Minor mask revision

68.4.1.4 Port Configuration Register (PCR)

The PCR is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

Note: The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

Register index: 127

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FPM	MCKO_GT	MCKO_EN	MCKO_DIV[2:0]			EVT_EN	DDR_EN	Reserved	0	0	0	Reserved	Reserved	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LP_DBG_EN	0	0	0	0	0	LP2_SYN	LP1_SYN	0	0	0	0	0	0	0	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1390. Port Configuration Register (PCR)

Table 1360. PCR field descriptions

Name	Description
0 FPM	Full Port Mode The value of the FPM bit determines if the auxiliary output port uses the full MDO port or a reduced MDO port to transmit messages. 1 All MDO pins are used to transmit messages 0 A subset of MDO pins are used to transmit messages
1 MCKO_GT	MCKO Clock Gating Control This bit is used to enable or disable MCKO clock gating. If clock gating is enabled, the MCKO clock is gated when the NPC is in enabled mode but not actively transmitting messages on the auxiliary output port. When clock gating is disabled, MCKO is allowed to run even if no auxiliary output port messages are being transmitted. 1 MCKO gating is enabled. 0 MCKO gating is disabled.
2 MCKO_EN	MCKO Enable This bit enables the MCKO clock to run. When enabled, the frequency of MCKO is determined by the MCKO_DIV field. 1 MCKO clock is enabled. 0 MCKO clock is driven to zero.
3:5 MCKO_DIV[2:0]	MCKO Division Factor The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. *Table 1361* shows the meaning of MCKO_DIV values. In this table, SYS_CLK represents the system clock frequency.
6 EVT_EN	$\overline{EVTO}$/$\overline{EVTI}$ Enable This bit enables the $\overline{EVTO}$/$\overline{EVTI}$ port functions. 0 $\overline{EVTO}$/$\overline{EVTI}$ port disabled 1 EVTO/EVTI port enabled

Table 1360. PCR field descriptions (continued)

Name	Description
7 DDR_EN	Double Data Rate Mode Enable This bit enables Nexus Double Data Rate (DDR) mode. In DDR mode, message data is updated on both rising and falling edges of MCKO, effectively doubling message throughput. 1 DDR mode enabled 0 DDR mode disabled
16 LP_DBG_EN	Low Power Debug Enable This bit enables debug functionality on exit from low power modes on supported devices. 1 Low power debug enabled 0 Low power debug disabled
22 LP2_SYN	Low Power Mode 2 Synchronization This bit is used to synchronize the entry into low power mode between the device and debug tool. Supported devices set this bit before a pending entry into low power mode. After reading the bit as set, the debug tool then clears the bit to acknowledge to the device that it may enter the low power mode. 1 Low power mode entry pending 0 Low power mode entry acknowledged
23 LP1_SYN	Low Power Mode 1 Synchronization This bit is used to synchronize the entry into low power mode between the device and debug tool. Supported devices set this bit before a pending entry into low power mode. After reading the bit as set, the debug tool then clears the bit to acknowledge to the device that it may enter the low power mode. 1 Low power mode entry pending 0 Low power mode entry acknowledged

Table 1361. PCR[MCKO_DIV] values

MCKO_DIV[2:0]	MCKO frequency
0d (000b)	SYS_CLK[1]
1d (001b)	SYS_CLK/2
2d (010b)	SYS_CLK/3[2]
3d (011b)	SYS_CLK/4
4d (100b)	Reserved
5d (101b)	Reserved
6d (110b)	Reserved
7d (111b)	SYS_CLK/8

1. The SYS_CLK setting for MCKO frequency should only be used if this setting does not violate the maximum operating frequency of the auxiliary port pins.
2. SYS_CLK/3 setting must not be used in DDR mode.

68.5 Functional description

68.5.1 NPC reset configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the auxiliary port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

The following table describes the NPC reset configuration options.

Table 1362. NPC reset configuration options

NPC_PCR[MCKO_EN]	NPC_PCR[FPM]	Configuration
X	X	Reset
0	X	Disabled
1	1	Full-Port mode
1	0	Reduced-Port mode

68.5.2 Auxiliary output port

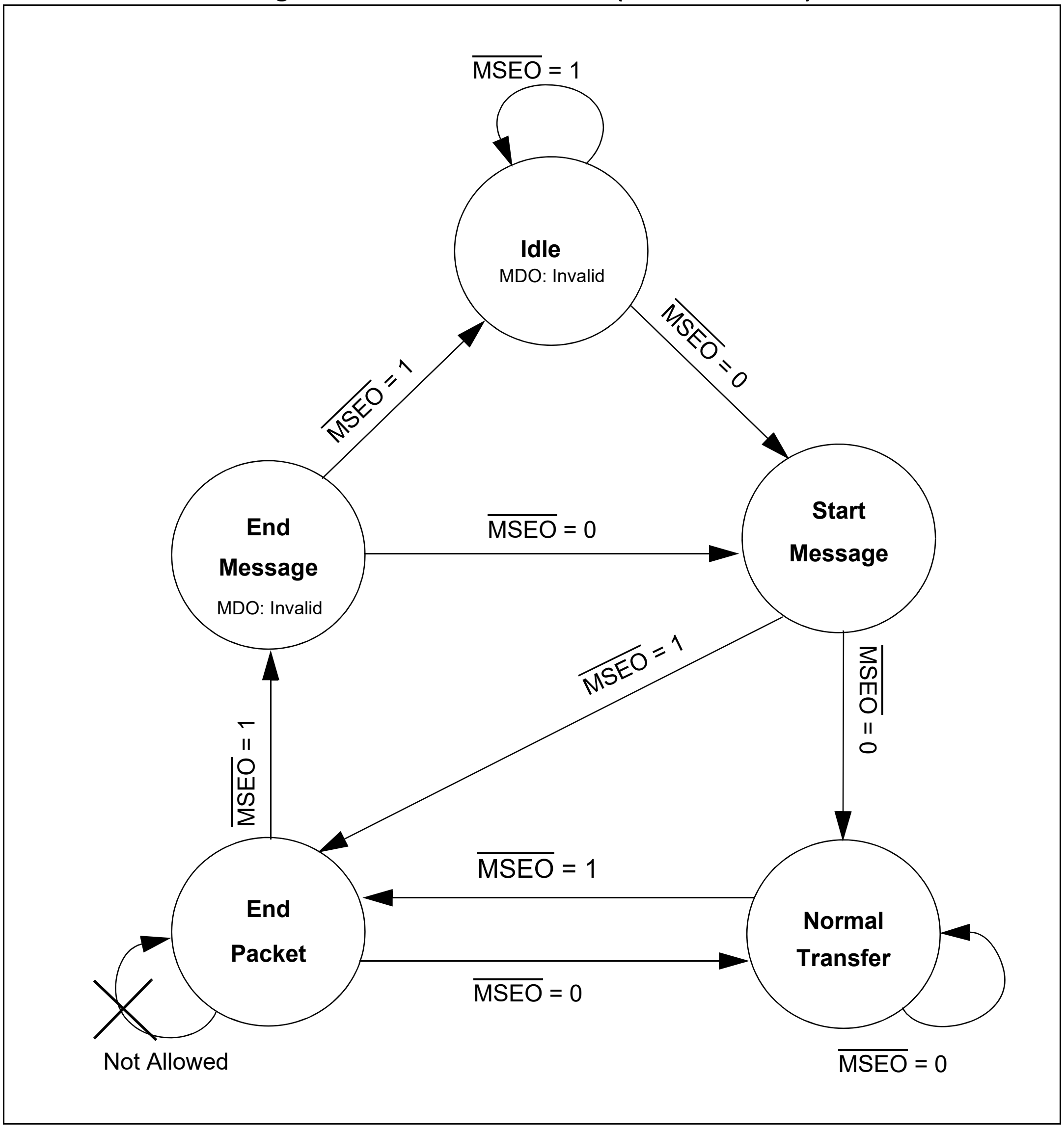
The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the Nexus modules and arbitrates for access to the port.

68.5.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the MSEO functions. The MSEO pin is used to signal the end of variable-length packets and the end of messages. It is not required to indicate the end of fixed-length packets. MDO and MSEO are sampled on the rising edge of MCKO.

Figure 1391 illustrates the state diagram for MSEO transfers. All transitions not included in the figure are reserved, and must not be used.

Figure 1391. $\overline{\text{MSEO}}$ transfers (for 1-bit $\overline{\text{MSEO}}$)



68.5.2.2 Output messages

In addition to sending out messages generated in other Nexus blocks, the NPC can also output the device ID message contained in the device ID register and the port replacement output message on the MDO pins. The device ID message can also be sent out serially through TDO.

Table 1363 describes the device ID and port replacement output messages that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Table 1363. NPC output messages

Message name	Packet				
	Size (bits)		Type	Name	Description
	Min	Max			
Device ID Message	6	6	fixed	TCODE	Value = 1
	32	32	fixed	ID	DID register contents

Figure 1392 shows the various message formats that the pin interface formatter has to encounter. Note that for variable-length fields, the transmitted size of the field is determined from the range of the least significant bit to the most significant non-zero-valued bit (that is most significant zero-valued bits are not transmitted).

Figure 1392. Message field sizes

Message	TCODE	Field #1	Field #2	Field #3	Field #4	Field #5	Size (bits) Min(1)	Size (bits) Max(2)
Device ID Message	1	Fixed = 32	—	—	—	—	38	38

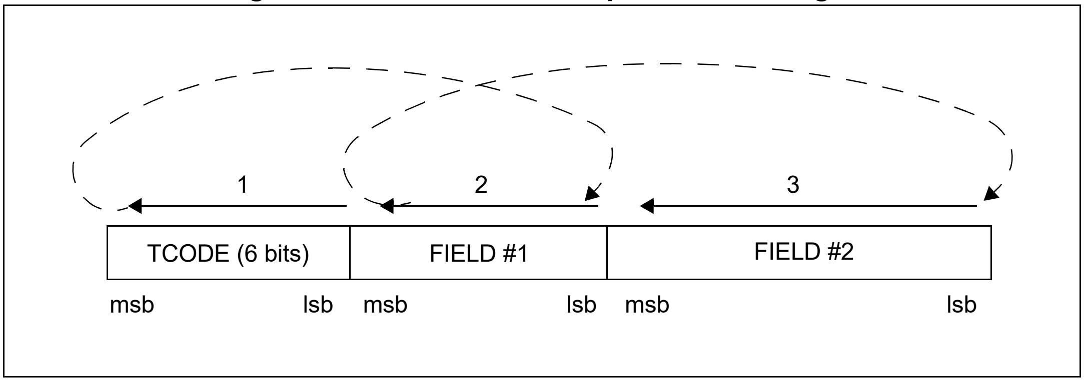
1. Minimum information size. The actual number of bits transmitted depends on the number of MDO pins.
2. Maximum information size. The actual number of bits transmitted depends on the number of MDO pins.

The double edges in *Figure 1392* indicate the starts and ends of messages. Fields without shaded areas between them are grouped into super-fields and can be transmitted together without end-of-packet indications between them.

68.5.2.3 Rules of message

- A variable-sized field within a message must end on a port boundary. (Port boundaries depend on the number of MDO pins active with the current reset configuration.)
- A variable-sized field may start within a port boundary only when following a fixed-length field.
- Super-fields must end on a port boundary.
- When a variable-length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary.
- Multiple fixed-length packets may start and/or end on a single clock.
- When any packet follows a variable-length packet, it must start on a port boundary.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information.
- Within a field, the lowest significant bits are shifted out first. *Figure 1393* shows the transmission sequence of a message that is made up of a TCODE followed by two fields.

Figure 1393. Transmission sequence of messages



68.5.3 IEEE 1149.1-2001 (JTAG) TAP

The NPC block uses the IEEE 1149.1-2001 TAP for accessing registers. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. There may also be other blocks on the MCU that use the TAP and implement a TAP controller.

Refer to the IEEE 1149.1-2001 specification for further details on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1-2001 state machine shown in *Figure 1395*. The Nexus controller state machine is defined by the IEEE-ISTO 5001-2001 standard. It is shown in *Figure 1396*.

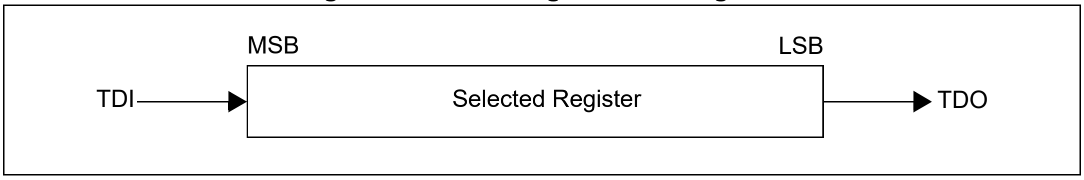
The instructions implemented by the NPC TAP controller are listed in *Table 1364*. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4 bits.

Table 1364. Implemented instructions

Instruction name	Private/Public	Opcode	Description
NEXUS-ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in *Figure 1394*. This applies for the instruction register and all Nexus tool-mapped registers.

Figure 1394. Shifting data into register

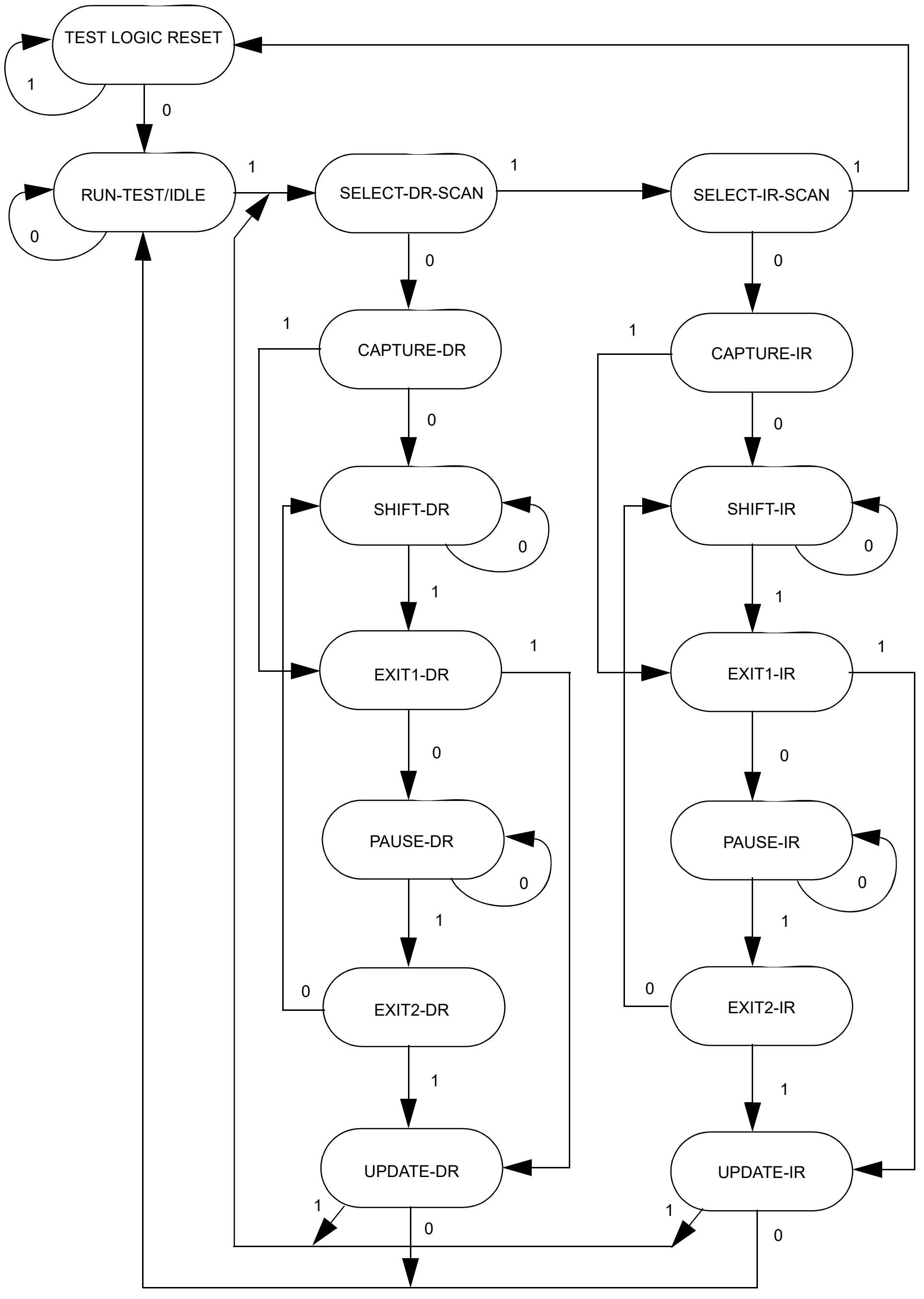


68.5.3.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal or entry into censored mode resets the NPC TAP controller.

When not in power-on reset or censored mode, the NPC TAP controller is enabled exiting the Test-Logic-Reset state loading the NEXUS-ENABLE instruction in NPC TAP controller.

Figure 1395. IEEE 1149.1-2001 TAP controller state machine



Note: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

68.5.3.2 Retrieving device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the auxiliary output port or shifted out serially by accessing the Nexus Device ID register through the TAP. Transmission of the device identification message on the auxiliary output port MDO pins occurs immediately after a write to the PCR, if the NPC is enabled. Transmission of the device identification message serially via TDO is achieved by performing a read of the register contents as described in *Section 68.5.3.4: Selecting a Nexus client register*.

68.5.3.3 Loading NEXUS-ENABLE instruction

Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in *Figure 1396*, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. *Table 1365* illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Figure 1396. NEXUS controller state machine

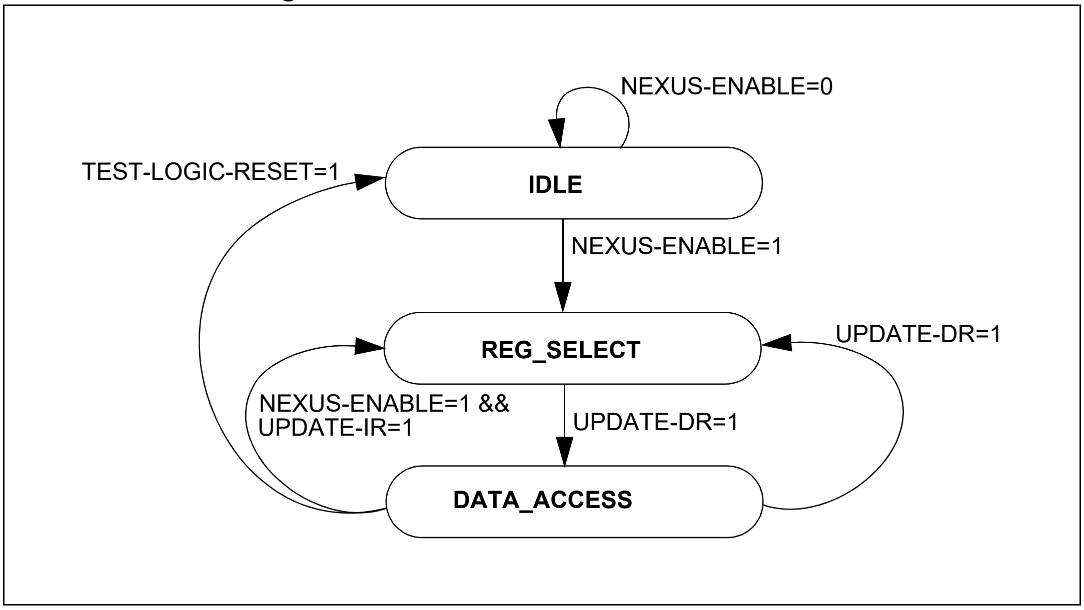


Table 1365. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
3 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in

Table 1365. Loading NEXUS-ENABLE instruction (continued)

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

68.5.3.4 Selecting a Nexus client register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in *Figure 1397*. The read/write control bit is set to '1' for writes and '0' for reads.

Figure 1397. IEEE 1149.1 controller command input

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 1366. Write to a 32-bit Nexus Client Register

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register

Table 1366. Write to a 32-bit Nexus Client Register (continued)

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
48	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
49	1	UPDATE-DR	DATA_ACCESS	Value written to register
50	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

68.5.4 Nexus JTAG port sharing

Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register.

68.5.5 MCKO and ipg_sync_mcko

MCKO is an output clock to the development tools used for the timing of $\overline{\text{MSEO}}$ and MDO pin functions. MCKO is derived from the system clock and its frequency is determined by the value of the MCKO_DIV field in the PCR. Possible operating frequencies include system clock, one-half system clock, one-quarter system clock, and one-eighth system clock speed.

The NPC also generates an MCKO clock gating control output signal. This output can be used by the MCKO generation logic to gate the transmission of MCKO when the auxiliary port is enabled but not transmitting messages. The setting of the MCKO_GT bit inside the PCR determines whether or not MCKO gating control is active. The MCKO_GT bit resets to a logic 0. In this state gating of MCKO is disabled. To enable gating of MCKO, the MCKO_GT bit in the PCR is written to a logic 1.

68.5.6 $\overline{\text{EVTO}}$ sharing

The NPC block controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. The NPC assumes incoming $\overline{\text{EVTO}}$ signals will be asserted for one system clock period. After receiving a single clock period of asserted $\overline{\text{EVTO}}$ from any Nexus client, the NPC latches the result, and drives $\overline{\text{EVTO}}$ for one MCKO period on the following clock. When there is no active MCKO, such as in disabled mode, the NPC drives $\overline{\text{EVTO}}$ for two system clock periods. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

68.6 Initialization/Application information

68.6.1 Accessing NPC tool-mapped registers

To initialize the TAP for Nexus register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Load the TAP controller with the NEXUS-ENABLE instruction

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.
2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.
2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE-ISTO 5001-2003 standard for more details.

69 Nexus handshake module (NPC_HNDSHK)

This module enables debug entry/exit across low-power modes (Stop, Halt, Standby).

The NPC_HNDSHK supports:

- Setting and clearing of the NPC PCR sync bit on low-power mode entry and exit
- Putting the core into debug mode on low-power mode exit
- Generating a falling edge on the JTAG TDO pad on low-power mode exit

69.1 Entry into STOP/STANDBY from RUN modes

For the device to be in debug mode on low-power exit, the debugger should set the LP_DBG and LP1_SYNC bit in the NPC PCR register.

On STOP, STANDBY entry, the MC_ME asserts the lp_mode_entry_req input after the clock disable process has completed and before the processor enters its stopped state. The mode transition will then not proceed until the lp_mode_entry_ack output has been asserted.

The notification to the debugger of a low-power mode entry consists in setting the NPC PCR [LP1_SYNC] bit by the NPC Handshake module.

The debugger acknowledges that the transition into a low-power mode may proceed by clearing NPC PCR[LP1_SYNC] bit.

The device enters into the low-power modes. In anticipation of the low-power mode exit notification, the TDO pad is driven to '1'.

69.2 Exit from STOP/STANDBY into RUN modes

On STOP mode exit, the MC_ME asserts the lp_mode_exit_req input after ensuring that the regulator and memories are in normal mode and before the processor exits its stopped state. The mode transition will then not proceed until the lp_mode_exit_req has been acknowledged. The MC_RGM asserts the exit_from_standby input when executing a reset sequence due to a STANDBY exit. The reset sequence will then not complete until the lp_mode_exit_req has been acknowledged.

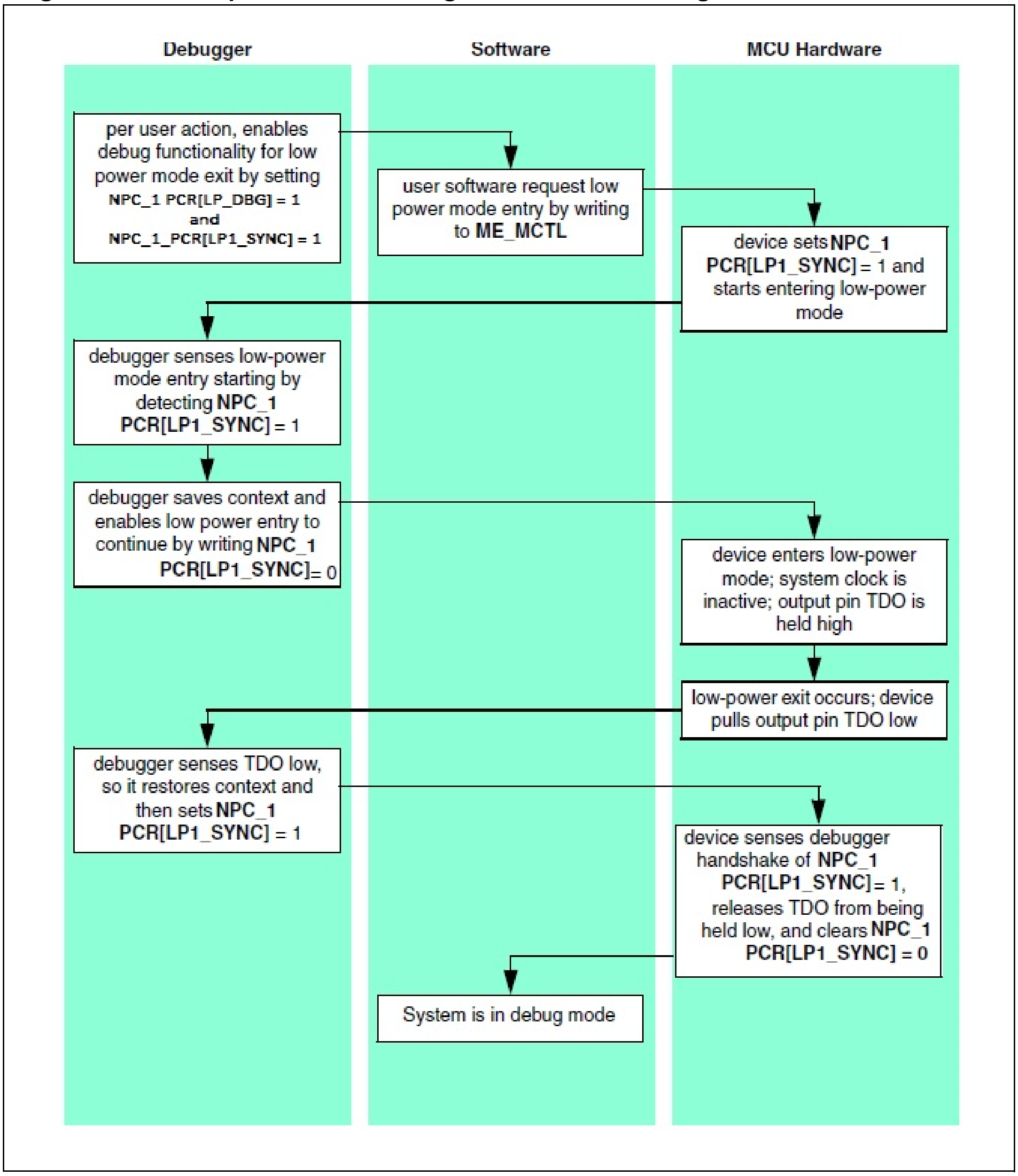
The notification to the debugger of a low-power mode exit consists in driving the TDO pad to '0'.

The debugger acknowledges that the transition from a low-power mode can continue by setting the NPC PCR[LP1_SYNC] bit.

Further, the NPC PCR[LP1_sync] bit is de-asserted by the hardware and the lp_mode_exit_ack output is sent to the MC_ME.

Note: The debugger clock multiplexer may not guarantee glitch free switching. Therefore, TCK should be disabled from when the debugger clears the sync bit in ENTRY_CLR until the debugger senses the falling edge of TDO in TDO_SET.

Figure 1398. Low-power mode debug handshake flow diagram for STOP, STANDBY



70 e200z420n3 Nexus 3 Module

The e200z420n3 Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port that is used in conjunction with the JTAG port in these processors.

70.1 Introduction

70.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

70.1.2 Terms and Definitions

Table 1367 contains a set of terms and definitions associated with the Nexus 3 module.

Table 1367. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design.
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.

Table 1367. Terms and Definitions (continued)

Term	Description
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	“System-on-a-Chip”. SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase “according to the standard” is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

70.1.3 Feature List

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (such as direct and indirect branches, exceptions), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.

- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging
- Auxiliary interface for higher data input/output
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
  - One (1) Read/Write Error pin (**nex_err_b**) pin
  - One (1) Watchpoint Event output pin (**nex_evto_b**)
  - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
  - One (1) Event In pin (**nex_evti_b**)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)

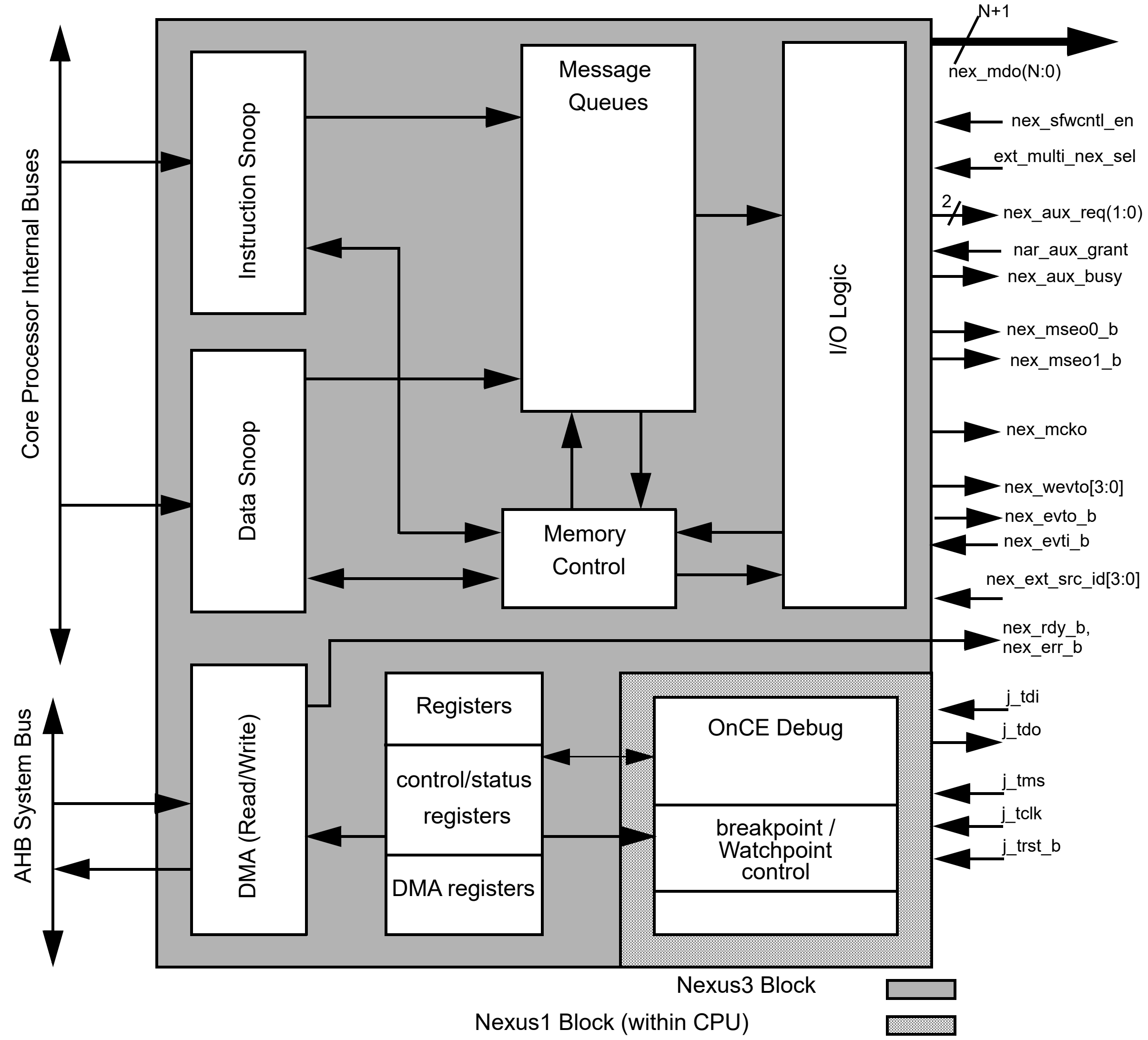
Note: For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM — maximum number of MDO pins) or Reduced Port Mode (RPM — minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note: The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

70.1.4 Functional Block Diagram

Figure 1399. Nexus 3 functional block diagram



Note: The “nex_aux_req[1:0]”, “nar_aux_grant” & “nex_aux_busy” signals are used for inter-module communication in a multi-Nexus environment. They are not pins on the SoC.

70.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (NEXUS3-ACCESS) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module’s DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU’s **p_reset_b** signal, even when software has control of the module.

70.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

70.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in *Table 1368*.

Table 1368. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace — Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace — Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
Data Trace — Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 1373*)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace — Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 1373*)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)

Table 1368. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace — Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace — Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace — Indirect Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	6	6	TCODE	fixed	TCODE number = 13
Data Trace — Data Write Message w/ Sync	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 1373*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table 1368. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Trace — Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 1373*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to *Table 1371*) — indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See *Section 70.12.4: Resource Full Messages*.)
Program Trace — Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	1	32	HIST	variable	branch / predicate instruction history (see *Section 70.12.1: Branch Trace Messaging types*)
Program Trace — Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See *Section 70.12.1: Branch Trace Messaging types*.)

Table 1368. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Program Trace — Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to *Table 1372*)
	2	2	CDF	fixed	# fields of information in CDATA. 00 — reserved, 01 — one field (CDATA1) (reserved), 10 — two fields (CDATA1 + CDATA2), 11 — three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 — [branch / predicate instruction history] (See *Section 70.12.5: Program Correlation Messages*.)
	0	32	CDATA2	variable	correlation data field 2— PID info (See *Section 70.12.5: Program Correlation Messages*.)

Note: Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Section 70.12.1: Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 1369 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 1369. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (such as Debug Status messages)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 1370 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 1370. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 – 1111	Reserved

Table 1371 shows the encodings used for resource codes for certain messages.

Table 1371. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 1372 shows the event code encodings used for certain messages.

Table 1372. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010–0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110–1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM} = 0 and $DC4_{PTMARK}$= 1
1010	Branch and link occurrence (direct branch function call)
1011–1111	Reserved for future functionality

Table 1373 shows the data trace size encodings used for certain messages.

Table 1373. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 — no data
0001	Byte

Table 1373. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14) (continued)

DTM Size Encoding	Transfer Size
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001–1111	Reserved

70.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See *Section 70.5: Nexus 3 Register Access via JTAG/OnCE* and *Section 70.6: Nexus 3 Register Access via Software* for details on Nexus 3 register access.

Note: Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

Table 1374 details the register map for the Nexus 3 module.

Table 1374. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #(1)
Client Select Control (CSC)(2)	0x1	R	0x02	—	
Port Configuration Register (PCR)(3)	PCR_INDEX(2)	R/W	—	—	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	—
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	—
Reserved	0x8	R/W	0x18	0x19	—
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	—
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	—
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	—
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377

Table 1374. Nexus 3 Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #(1)
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 → 0x2F	—	0x28→0x5E	0x29→0x5F	—
Development Status (DS)	0x30	R	0x60	—	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	—	0x68	0x69	—
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 → 0x3F	—	0x72→0x7E	0x73→0x7F	—

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

70.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.

7	6	5	4	3	2	1	0
Reserved				CS			

Nexus Reg# 0x1; Read-only; Reset: 0x0

Figure 1400. Client Select Control (CSC) register

Table 1375. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0*xx* Nexus client (SoC level)

70.4.2 Port Configuration Register (PCR) — reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPC	0	MCK_EN	MCK_DIV			0																									

Nexus Reg# PCR_INDEX; Read/Write; Reset: 0x0

Figure 1401. Port Configuration Register (PCR)

Table 1376. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note: The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

70.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in *Figure 1402* and its fields are described in *Table 1377*.

<table>
<tr><td>OPC</td><td colspan="2">MCK_DIV</td><td>0</td><td>PTM</td><td colspan="2">0</td><td>OTS</td><td colspan="9">0</td><td>POTD</td><td colspan="2">TSEN</td><td colspan="2">EOC</td><td colspan="2">EIC</td><td colspan="2">0</td><td colspan="6">TM</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Nexus Reg# 0x2; DCR: 368; Read/Write; Reset: 0x0

Figure 1402. Development Control 1 (DC1) register

Table 1377. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 — Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 — Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 — **nex_mcko** is 1x processor clock freq. 01 — **nex_mcko** is 1/2x processor clock freq. 10 — **nex_mcko** is 1/4x processor clock freq. 11 — **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 — Program Trace uses traditional Branch Messages 1 — Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 — PID0 data is transmitted within Ownership Trace Messages 1 — Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 — Periodic Ownership Trace message events are enabled 1 — Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable — (not implemented, write to 00) 00 — Timestamp is disabled
11:10	EOC	EVTO Control 00 — **nex_evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 — **nex_evto_b** upon entry into Debug Mode 1*x* — Reserved
9:8	EIC	EVTI Control 00 — **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 — **nex_evti_b** is used for Debug request 10 — **nex_evti_b** is disabled 1*x* — Reserved

Table 1377. DC1 field descriptions (continued)

Bits	Name	Description
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode(1) 000000 — All Trace Disabled *xxxxx*1 — Ownership Trace enabled *xxxx*1*x* — Data Trace enabled *xxx*1*xx* — Program Trace enabled *xx*1*xxx* — Watchpoint Trace enabled *x*1*xxxx* — Reserved 1*xxxxx* — Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enabled for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to *Section 70.4.7: Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers* for more information on watchpoint triggering. Refer to *Section 70.16: External Hardware Trigger Controls* for more information on external hardware triggering.

Note: The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

70.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z420n3) Core Debug Support chapter.

Development Control Register 2 is shown in *Figure 1403* and its fields are described in *Table 1378*.

31–24	23–16	15–8	7–0
WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WEVTO[3]C				WEVTO[2]C				WEVTO[1]C				WEVTO[0]C				EWC															

Nexus Reg# 0x3; DCR: 369; Read/Write; Reset: 0x0

Figure 1403. Development Control 2 (DC2) register

Table 1378. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[3]** 0001 — Watchpoint #0 triggers **nex_wevto[3]** 0010 — Watchpoint #1 triggers **nex_wevto[3]** 0011 — Watchpoint #2 triggers **nex_wevto[3]** 0100 — Watchpoint #3 triggers **nex_wevto[3]** 0101 — Watchpoint #4 triggers **nex_wevto[3]** 0110 — Watchpoint #5 triggers **nex_wevto[3]** 0111 — Watchpoint #6 triggers **nex_wevto[3]** 1000 — Watchpoint #7 triggers **nex_wevto[3]** 1001 — Watchpoint #8 triggers **nex_wevto[3]** 1010 — Watchpoint #9 triggers **nex_wevto[3]** 1011 — Watchpoint #10 triggers **nex_wevto[3]** 1100 — Watchpoint #11 triggers **nex_wevto[3]** 1101 — Watchpoint #12 triggers **nex_wevto[3]** 1110 — Watchpoint #13 triggers **nex_wevto[3]** 1111 — Watchpoint #14 triggers **nex_wevto[3]**
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[2]** 0001 — Watchpoint #0 triggers **nex_wevto[2]** 0010 — Watchpoint #1 triggers **nex_wevto[2]** 0011 — Watchpoint #2 triggers **nex_wevto[2]** 0100 — Watchpoint #3 triggers **nex_wevto[2]** 0101 — Watchpoint #4 triggers **nex_wevto[2]** 0110 — Watchpoint #5 triggers **nex_wevto[2]** 0111 — Watchpoint #6 triggers **nex_wevto[2]** 1000 — Watchpoint #7 triggers **nex_wevto[2]** 1001 — Watchpoint #8 triggers **nex_wevto[2]** 1010 — Watchpoint #9 triggers **nex_wevto[2]** 1011 — Watchpoint #10 triggers **nex_wevto[2]** 1100 — Watchpoint #11 triggers **nex_wevto[2]** 1101 — Watchpoint #12 triggers **nex_wevto[2]** 1110 — Watchpoint #13 triggers **nex_wevto[2]** 1111 — Watchpoint #14 triggers **nex_wevto[2]**
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[1]** 0001 — Watchpoint #0 triggers **nex_wevto[1]** 0010 — Watchpoint #1 triggers **nex_wevto[1]** 0011 — Watchpoint #2 triggers **nex_wevto[1]** 0100 — Watchpoint #3 triggers **nex_wevto[1]** 0101 — Watchpoint #4 triggers **nex_wevto[1]** 0110 — Watchpoint #5 triggers **nex_wevto[1]** 0111 — Watchpoint #6 triggers **nex_wevto[1]** 1000 — Watchpoint #7 triggers **nex_wevto[1]** 1001 — Watchpoint #8 triggers **nex_wevto[1]** 1010 — Watchpoint #9 triggers **nex_wevto[1]** 1011 — Watchpoint #10 triggers **nex_wevto[1]** 1100 — Watchpoint #11 triggers **nex_wevto[1]** 1101 — Watchpoint #12 triggers **nex_wevto[1]** 1110 — Watchpoint #13 triggers **nex_wevto[1]** 1111 — Watchpoint #14 triggers **nex_wevto[1]**

Table 1378. DC2 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[0]** 0001 — Watchpoint #0 triggers **nex_wevto[0]** 0010 — Watchpoint #1 triggers **nex_wevto[0]** 0011 — Watchpoint #2 triggers **nex_wevto[0]** 0100 — Watchpoint #3 triggers **nex_wevto[0]** 0101 — Watchpoint #4 triggers **nex_wevto[0]** 0110 — Watchpoint #5 triggers **nex_wevto[0]** 0111 — Watchpoint #6 triggers **nex_wevto[0]** 1000 — Watchpoint #7 triggers **nex_wevto[0]** 1001 — Watchpoint #8 triggers **nex_wevto[0]** 1010 — Watchpoint #9 triggers **nex_wevto[0]** 1011 — Watchpoint #10 triggers **nex_wevto[0]** 1100 — Watchpoint #11 triggers **nex_wevto[0]** 1101 — Watchpoint #12 triggers **nex_wevto[0]** 1110 — Watchpoint #13 triggers **nex_wevto[0]** 1111 — Watchpoint #14 triggers **nex_wevto[0]**
15:0	EWC	EVTO Watchpoint Configuration(1) 0000000000000000 No Watchpoints #0–15 trigger **nex_evto_b** *xxxxxxxxxxxxxxx*1 Watchpoint #0 triggers **nex_evto_b** *xxxxxxxxxxxxxx*1*x* Watchpoint #1 triggers **nex_evto_b** *xxxxxxxxxxxxx*1*xx* Watchpoint #2 triggers **nex_evto_b** *xxxxxxxxxxxx*1*xxx* Watchpoint #3 triggers **nex_evto_b** *xxxxxxxxxxx*1*xxxx* Watchpoint #4 triggers **nex_evto_b** *xxxxxxxxxx*1*xxxxx* Watchpoint #5 triggers **nex_evto_b** *xxxxxxxxx*1*xxxxxx* Watchpoint #6 triggers **nex_evto_b** *xxxxxxxx*1*xxxxxxx* Watchpoint #7 triggers **nex_evto_b** *xxxxxxx*1*xxxxxxxx* Watchpoint #8 triggers **nex_evto_b** *xxxxxx*1*xxxxxxxxx* Watchpoint #9 triggers **nex_evto_b** *xxxxx*1*xxxxxxxxxx* Watchpoint #10 triggers **nex_evto_b** *xxxx*1*xxxxxxxxxxx* Watchpoint #11 triggers **nex_evto_b** *xxx*1*xxxxxxxxxxxx* Watchpoint #12 triggers **nex_evto_b** *xx*1*xxxxxxxxxxxxx* Watchpoint #13 triggers **nex_evto_b** *x*1*xxxxxxxxxxxxxx* Watchpoint #14 triggers **nex_evto_b** 1*xxxxxxxxxxxxxxx* Watchpoint #15 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in *Figure 1404* and its fields are described in *Table 1379*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WEVTO[3]C				WEVTO[2]C				WEVTO[1]C				WEVTO[0]C				0		EWC													

Nexus Reg# 0x4; DCR: 370; Read/Write; Reset: 0x0

Figure 1404. Development Control 3 (DC3) register

Table 1379. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #15–#26 trigger **nex_wevto[3]** 0001 Watchpoint #15 triggers **nex_wevto[3]** 0010 Watchpoint #16 triggers **nex_wevto[3]** 0011 Watchpoint #17 triggers **nex_wevto[3]** 0100 Watchpoint #18 triggers **nex_wevto[3]** 0101 Watchpoint #19 triggers **nex_wevto[3]** 0110 Watchpoint #20 triggers **nex_wevto[3]** 0111 Watchpoint #21 triggers **nex_wevto[3]** 1000 Watchpoint #22 triggers **nex_wevto[3]** 1001 Watchpoint #23 triggers **nex_wevto[3]** 1010 Watchpoint #24 triggers **nex_wevto[3]** 1011 Watchpoint #25 triggers **nex_wevto[3]** 1100 Watchpoint #26 triggers **nex_wevto[3]** 11*x*1 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #15–#26 trigger **nex_wevto[2]** 0001 Watchpoint #15 triggers **nex_wevto[2]** 0010 Watchpoint #16 triggers **nex_wevto[2]** 0011 Watchpoint #17 triggers **nex_wevto[2]** 0100 Watchpoint #18 triggers **nex_wevto[2]** 0101 Watchpoint #19 triggers **nex_wevto[2]** 0110 Watchpoint #20 triggers **nex_wevto[2]** 0111 Watchpoint #21 triggers **nex_wevto[2]** 1000 Watchpoint #22 triggers **nex_wevto[2]** 1001 Watchpoint #23 triggers **nex_wevto[2]** 1010 Watchpoint #24 triggers **nex_wevto[2]** 1011 Watchpoint #25 triggers **nex_wevto[2]** 1100 Watchpoint #26 triggers **nex_wevto[2]** 11*x*1 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #15–#29 trigger **nex_wevto[1]** 0001 Watchpoint #15 triggers **nex_wevto[1]** 0010 Watchpoint #16 triggers **nex_wevto[1]** 0011 Watchpoint #17 triggers **nex_wevto[1]** 0100 Watchpoint #18 triggers **nex_wevto[1]** 0101 Watchpoint #19 triggers **nex_wevto[1]** 0110 Watchpoint #20 triggers **nex_wevto[1]** 0111 Watchpoint #21 triggers **nex_wevto[1]** 1000 Watchpoint #22 triggers **nex_wevto[1]** 1001 Watchpoint #23 triggers **nex_wevto[1]** 1010 Watchpoint #24 triggers **nex_wevto[1]** 1011 Watchpoint #25 triggers **nex_wevto[1]** 1100 Watchpoint #26 triggers **nex_wevto[1]** 1101 Watchpoint #27 triggers **nex_wevto[1]** 1110 Watchpoint #28 triggers **nex_wevto[1]** 1111 Watchpoint #29 triggers **nex_wevto[1]**

Table 1379. DC3 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 No Watchpoints #15–#29 trigger **nex_wevto[0]** 0001 Watchpoint #15 triggers **nex_wevto[0]** 0010 Watchpoint #16 triggers **nex_wevto[0]** 0011 Watchpoint #17 triggers **nex_wevto[0]** 0100 Watchpoint #18 triggers **nex_wevto[0]** 0101 Watchpoint #19 triggers **nex_wevto[0]** 0110 Watchpoint #20 triggers **nex_wevto[0]** 0111 Watchpoint #21 triggers **nex_wevto[0]** 1000 Watchpoint #22 triggers **nex_wevto[0]** 1001 Watchpoint #23 triggers **nex_wevto[0]** 1010 Watchpoint #24 triggers **nex_wevto[0]** 1011 Watchpoint #25 triggers **nex_wevto[0]** 1100 Watchpoint #26 triggers **nex_wevto[0]** 1101 Watchpoint #27 triggers **nex_wevto[0]** 1110 Watchpoint #28 triggers **nex_wevto[0]** 1111 Watchpoint #29 triggers **nex_wevto[0]**
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration(1) 00000000000000 No Watchpoints #16–#29 trigger **nex_evto_b** *xxxxxxxxxxxxx*1 Watchpoint #16 triggers **nex_evto_b** *xxxxxxxxxxxx*1*x* Watchpoint #17 triggers **nex_evto_b** *xxxxxxxxxxx*1*xx* Watchpoint #18 triggers **nex_evto_b** *xxxxxxxxxx*1*xxx* Watchpoint #19 triggers **nex_evto_b** *xxxxxxxxx*1*xxxx* Watchpoint #20 triggers **nex_evto_b** *xxxxxxxx*1*xxxxx* Watchpoint #21 triggers **nex_evto_b** *xxxxxxx*1*xxxxxx* Watchpoint #22 triggers **nex_evto_b** *xxxxxx*1*xxxxxxx* Watchpoint #23 triggers **nex_evto_b** *xxxxx*1*xxxxxxxx* Watchpoint #24 triggers **nex_evto_b** *xxxx*1*xxxxxxxxx* Watchpoint #25 triggers **nex_evto_b** *xxx*1*xxxxxxxxxx* Watchpoint #26 triggers **nex_evto_b** *xx*1*xxxxxxxxxxx* Watchpoint #27 triggers **nex_evto_b** *x*1*xxxxxxxxxxxx* Watchpoint #28 triggers **nex_evto_b** 1*xxxxxxxxxxxxx* Watchpoint #29 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

70.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in *Figure 1405* and its fields are described in *Table 1380*.

PTMARK	DTMARK	0	EVCDM
31	30	29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x5; DCR: 371; Read/Write; Reset: 0x0

Figure 1405. Development Control 4 (DC4) register

Table 1380. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask(1) 0000000000000000 No EVCODEs masked for Program Correlation Messages *xxxxxxxxxxxxxxx*1 EVCODE #0 is masked for Program Correlation Messages *xxxxxxxxxxxxxx*1*x* EVCODE #1 is masked for Program Correlation Messages *xxxxxxxxxxxxx*1*xx* EVCODE #2 is masked for Program Correlation Messages *xxxxxxxxxxxx*1*xxx* EVCODE #3 is masked for Program Correlation Messages *xxxxxxxxxxx*1*xxxx* EVCODE #4 is masked for Program Correlation Messages *xxxxxxxxxx*1*xxxxx* EVCODE #5 is masked for Program Correlation Messages *xxxxxxxxx*1*xxxxxx* EVCODE #6 is masked for Program Correlation Messages *xxxxxxxx*1*xxxxxxx* EVCODE #7 is masked for Program Correlation Messages *xxxxxxx*1*xxxxxxxx* EVCODE #8 is masked for Program Correlation Messages *xxxxxx*1*xxxxxxxxx* EVCODE #9 is masked for Program Correlation Messages *xxxxx*1*xxxxxxxxxx* EVCODE #10 is masked for Program Correlation Messages *xxxx*1*xxxxxxxxxxx* EVCODE #11 is masked for Program Correlation Messages *xxx*1*xxxxxxxxxxxx* EVCODE #12 is masked for Program Correlation Messages *xx*1*xxxxxxxxxxxxx* EVCODE #13 is masked for Program Correlation Messages *x*1*xxxxxxxxxxxxxx* EVCODE #14 is masked for Program Correlation Messages 1*xxxxxxxxxxxxxxx* EVCODE #15 is masked for Program Correlation Messages

1. Refer to *Table 1372* for implemented EVCODEs.

70.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.

DBG	LPS			LPC		0																									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x4; DCR: 409; Read-only; Reset: 0x0

Figure 1406. Development Status (DS) register

Table 1381. DS field descriptions

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting State (**p_waiting** signal asserted 010 Halted State (**p_halted** signal asserted 011 Reserved 1*xx* Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

70.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allow the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related “window” for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4’b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z420n3) Core Debug Support chapter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTS				PTE				DTS				DTE				0															

Nexus Reg# 0xB; DCR: 375; Read/Write; Reset: 0x0

Figure 1407. Watchpoint Trigger (WT) register

Table 1382 details the Watchpoint Trigger register fields.

Table 1382. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

PTST																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x35; DCR: 412; Read/Write; Reset: 0x0

Figure 1408. Program Trace Start Trigger Control (PTSTC) register

Table 1383 details the PTSTC register fields.

Table 1383. PTSTC field descriptions

Bits	Name	Description	
31:0	PTST	Program Trace Start Trigger Control	
		00000000000000000000000000000000	Trigger disabled
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1	Use Watchpoint #0
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x	Use Watchpoint #1
		xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx	Use Watchpoint #2
		xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx	Use Watchpoint #3
		xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx	Use Watchpoint #4
		xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx	Use Watchpoint #5
		xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx	Use Watchpoint #6
		xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Use Watchpoint #7
		xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx	Use Watchpoint #8
		xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx	Use Watchpoint #9
		xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx	Use Watchpoint #10
		xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Use Watchpoint #11
		xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx	Use Watchpoint #12
		xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx	Use Watchpoint #13
		xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx	Use Watchpoint #14
		xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx	Use Watchpoint #15
		xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx	Use Watchpoint #16
		xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx	Use Watchpoint #17
		xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx	Use Watchpoint #18
		xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx	Use Watchpoint #19
		xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxxx	Use Watchpoint #20
		xxxxxxxxxx1xxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #21
		xxxxxxxxx1xxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #22
		xxxxxxxx1xxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #23
		xxxxxxx1xxxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #24
		xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #25
		xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #26
		xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #27
		xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #28
		xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #29
		x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #30
		1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

PTET

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x36; DCR: 413; Read/Write; Reset: 0x0

Figure 1409. Program Trace End Trigger Control (PTETC) register

Table 1384 details the PTETC register fields.

Table 1384. PTETC field descriptions

Bits	Name	Description
31:0	PTET	Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx Use Watchpoint #16 xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx Use Watchpoint #17 xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Use Watchpoint #18 xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Use Watchpoint #19 xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxxx Use Watchpoint #20 xxxxxxxxxx1xxxxxxxxxxxxxxxxxxxxx Use Watchpoint #21 xxxxxxxxx1xxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #22 xxxxxxxx1xxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #23 xxxxxxx1xxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #24 xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #25 xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #26 xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #27 xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #28 xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #29 x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #30 1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		DTST																													

Nexus Reg# 0x37; DCR: 414; Read/Write; Reset: 0x0

Figure 1410. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 1385. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #26 *xx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #27 *x*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #28 1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		DTET																													

Nexus Reg# 038; DCR: 415; Read/Write; Reset: 0x0

Figure 1411. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 1386. DTETC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxx1xxxxxxxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxx1xxxxxxxxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxx1xxxxxxxxxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxx1xxxxxxxxxxxxxxx Use Watchpoint #15 xxxxxxxxxxxxx1xxxxxxxxxxxxxxxx Use Watchpoint #16 xxxxxxxxxxxx1xxxxxxxxxxxxxxxxx Use Watchpoint #17 xxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Use Watchpoint #18 xxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Use Watchpoint #19 xxxxxxxxx1xxxxxxxxxxxxxxxxxxxx Use Watchpoint #20 xxxxxxxx1xxxxxxxxxxxxxxxxxxxxx Use Watchpoint #21 xxxxxxx1xxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #22 xxxxxx1xxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #23 xxxxx1xxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #24 xxxx1xxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #25 xxx1xxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #26 xx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #27 x1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #28 1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #29

70.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

WEM																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# 0x33; DCR: 411; Read/Write; Reset: 0x0

Figure 1412. Watchpoint Mask (WMSK) register

Table 1387 details the Watchpoint Trigger register fields.

Table 1387. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Watchpoint #0 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Watchpoint #1 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Watchpoint #2 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Watchpoint #3 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Watchpoint #4 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Watchpoint #5 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Watchpoint #6 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Watchpoint #7 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Watchpoint #8 enabled for WTM xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Watchpoint #9 enabled for WTM xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Watchpoint #10 enabled for WTM xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #11 enabled for WTM xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx Watchpoint #12 enabled for WTM xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx Watchpoint #13 enabled for WTM xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx Watchpoint #14 enabled for WTM xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx Watchpoint #15 enabled for WTM xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx Watchpoint #16 enabled for WTM xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx Watchpoint #17 enabled for WTM xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Watchpoint #18 enabled for WTM xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #19 enabled for WTM xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxxx Watchpoint #20 enabled for WTM xxxxxxxxxx1xxxxxxxxxxxxxxxxxxxxx Watchpoint #21 enabled for WTM xxxxxxxxx1xxxxxxxxxxxxxxxxxxxxxx Watchpoint #22 enabled for WTM xxxxxxxx1xxxxxxxxxxxxxxxxxxxxxxx Watchpoint #23 enabled for WTM xxxxxxx1xxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #24 enabled for WTM xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #25 enabled for WTM xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #26 enabled for WTM xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #27 enabled for WTM xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #28 enabled for WTM xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #29 enabled for WTM x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #30 enabled for WTM 1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Watchpoint #31 enabled for WTM

70.4.9 Nexus Overrun Control (OVCR) register

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

31:30	29:28	27:22	21:16	15:14	13:12	11:1	0
0	SPTHOLD	0	SPEN	0	STTHOLD	0	STEN

Nexus Reg# 0x32; DCR: 410; Read/Write; Reset: 0x0

Figure 1413. Nexus Overrun Control (OVCR) register

Table 1388. OVCR field descriptions

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 Suppression threshold is when message queues are 1/4 full 01 Suppression threshold is when message queues are 1/2 full 10 Suppression threshold is when message queues are 3/4 full 11 Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 Suppression is disabled *xxxxx*1 Ownership Trace message suppression is enabled *xxxx*1*x* Data Trace message suppression is enabled *xxx*1*xx* Program Trace message suppression is enabled *xx*1*xxx* Watchpoint Trace message suppression is enabled *x*1*xxxx* Reserved 1*xxxxx* Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 Stall threshold is when message queues are 1/4 full 01 Stall threshold is when message queues are 1/2 full 10 Stall threshold is when message queues are 3/4 full 11 Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 Stalling is disabled 1 Stalling is enabled

70.4.10 Data Trace Control (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWT1		RWT2		RWT3		RWT4		0								STDC1	STDC2	STDC3	STDC4	0				RC1	RC2	RC3	RC4	DI	0		

Nexus Reg# 0xD; DCR: 376; Read/Write; Reset: 0x0

Figure 1414. Data Trace Control (DTC) register

The following table details the Data Trace Control register fields.

Table 1389. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs

Table 1389. DTC field descriptions (continued)

Bits	Name	Description
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses **Note:** The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. **Note:** The setting of the DI bit also affects the information provided on the data trace port outputs (See the “Data Trace Port Signals” section in the Core (e200z420n3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

70.4.11 Data Trace Start Address Registers (DTSA1–4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0xE; DCR: 377; Read/Write; Reset: 0x0

Figure 1415. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0xF; DCR: 378; Read/Write; Reset: 0x0

Figure 1416. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x10; DCR: 379; Read/Write; Reset: 0x0

Figure 1417. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x11; DCR: 380; Read/Write; Reset: 0x0

Figure 1418. Data Trace Start Address 4 (DTSA4) register

70.4.12 Data Trace End Address 1–4 (DTEA1–4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x12; DCR: 381; Read/Write; Reset: 0x0

Figure 1419. Data Trace End Address 1 register

Data Trace End Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x13; DCR: 382; Read/Write; Reset: 0x0

Figure 1420. Data Trace End Address 2 register

Data Trace End Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x14; DCR: 383; Read/Write; Reset: 0x0

Figure 1421. Data Trace End Address 3 register

Data Trace End Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x15; DCR: 408; Read/Write; Reset: 0x0

Figure 1422. Data Trace End Address 4 register

Table 1390 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 1390. Data trace — address range options

Programmed values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA → ← DTEA
DTSA < DTEA	1	← DTSA DTEA →
DTSA > DTEA	N/A	Invalid Range — no trace
DTSA = DTEA	N/A	Invalid Range — no trace

Note: DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are inclusive of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are exclusive of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI} = 1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (that is when the instruction prefetch is discarded).

70.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes. Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values. The RWCS Register also provides Read/Write Access Status information per *Table 1392*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AC	RW	SZ			MAP			PR		ATTR				0		CNT														ERR	DV

Nexus Reg# 0x7; Read/Write[(1)]; Reset: 0x0

Figure 1423. Read/Write Access Control/Status (RWCS) register

1. ERR and DV are read-only

Table 1391. RWCS field descriptions

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 1*xx* Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map All other values are reserved
RWCS[23:22]	PR[(1)]	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority

Table 1391. RWCS field descriptions (continued)

Bits	Name	Description
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR(2)	Read/Write Access Error (see *Table 1392*)
RWCS[0]	DV(2)	Read/Write Access Data Valid (see *Table 1392*)

1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 1392. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

70.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Read/Write Data
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0xA; Read/Write; Reset: 0x0

Figure 1424. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. *Table 1393* shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 1393. RWD data placement for transfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	*x x x*	0 0 0	—	—	—	X
Half	*x x* 0	0 0 1	—	—	X	X
Word	*x* 0 0	0 1 0	X	X	X	X
Doubleword first RWD pass (low order data) second RWD pass (high order data)	0 0 0	0 1 1	X X	X X	X X	X X

Table Notes:
"X" indicates byte lanes with valid data
"—" indicates byte lanes that will contain unused data.

Table 1394 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 1394. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	AHB[7:0]
Byte @001	0 0 1	—	—	—	AHB[15:8]
Byte @010	0 1 0	—	—	—	AHB[23:16]
Byte @011	0 1 1	—	—	—	AHB[31:24]
Byte @100	1 0 0	—	—	—	AHB[39:32]
Byte @101	1 0 1	—	—	—	AHB[47:40]
Byte @110	1 1 0	—	—	—	AHB[55:48]
Byte @111	1 1 1	—	—	—	AHB[63:56]
Half @000	0 0 0	—	—	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	—	—	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	—	—	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	—	—	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table 1394. RWD byte lane mapping (continued)

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Doubleword @000 first RWD pass second RWD pass	0 0 0	AHB[31:24] AHB[63:56]	AHB[23:16] AHB[55:48]	AHB[15:8] AHB[47:40]	AHB[7:0] AHB[39:32]

Table Notes:
"—" indicates byte lanes that will contain unused data.

70.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Read/Write Address
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# 0x9; Read/Write; Reset: 0x0

Figure 1425. Read/Write Access Address (RWA) register

70.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in *Table 1374*.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See *Section 70.23: IEEE 1149.1 (JTAG) RD/WR sequences*.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see *Table 1374*), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

Figure 1426. JTAG DR format

(7 bits)	(1 bit)
Nexus Register Index	R/W

RESET Value: 0x00

Table 1395. JTAG DR field descriptions

Nexus Register Index	Selected from values in *Table 1374*
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the “Capture-DR” state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the “Update-DR” state.

70.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these accesses only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. *Table 1374* shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

70.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

70.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

70.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

70.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant ‘1’. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

Figure 1427. Relative address generation and recreation

Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 1111 0110 0100

Address Message (M1) = 1111 0110 0100

Address Re-creation:

A1 ⊕ M1 = A2
A1 = 0000 0000 0000 0011 1111 1100 0000 0001
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

70.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

70.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

70.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

70.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to *Section 70.4.9: Nexus Overrun Control (OVCR) register*, for complete programming options.

70.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in

Section 70.8.1: Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

70.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstance:

- More than three messages are generated in the same cycle

Table 1396 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in *Table 1396*.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (newer instruction) (WPM → DQM → OTM → BTM → DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 1396. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A(1)
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A(1)
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM — PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM — PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message(2)

Table 1396. Message Type Priority and Message Dropped Responses (continued)

Message Type	Message	Priority	Message Dropped Response
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM — Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

70.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

70.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

70.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

70.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

70.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

70.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

Figure 1428. Debug Status message format

(8 bits)	(4 bits)	(6 bits)
DS[31:24]	Src. Proc.	TCODE (000000)

Fixed length = 18 bits

70.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note: The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see *Table 1369* and *Table 1370*):

Figure 1429. Error message format

(6 bits)	(4 bits)	(4 bits)	(6 bits)
Error Code	Error Type	Src. Proc.	TCODE (001000)

Fixed length = 20 bits

70.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

70.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

70.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z420n3) processors contain a *PowerISA 2.06* defined “Process ID” register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr**/**mtspr** instructions.

Note: The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr**/**mtspr** instructions. This register is accessible in user or supervisor mode.

Nexus 3 Process ID
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR — 517; Read/Write; Reset: 0

Figure 1430. Nexus 3 Process ID Register (NPIDR)

Note: OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will

indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

Figure 1431. Ownership Trace Message format

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15–46 bits

70.12 Program Trace

This section details the program trace mechanism supported by Nexus3 for the e200z420n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z420n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

70.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

70.12.1.1 Indirect Branch Message instructions

Table 1397 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 1397. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr**, **bcctrl**, **bclr**, **bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	se_sc, tw
Return from interrupts / exceptions	se_rfi, se_rfci, se_rfdi
Exit from reset with Program Trace Enabled	Indirect branch with Sync, target address is initial instruction, count = 1

70.12.1.2 Direct Branch Message Instructions

Table 1398 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 1398. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b**, **ba**, **bl**, **bla**, **bc**, **bca**, **bcl**, **bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

70.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

70.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of “events” that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

70.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages — Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages — Indirect Branch History, and Indirect Branch History with Synchronization Messages.

70.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

Figure 1432. Indirect branch message (history) format

(1-32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

70.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

Figure 1433. Indirect branch message format

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

70.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

Figure 1434. Direct branch message format

(1–8 bits)	(4 bits)	(6 bits)
Sequence Count	Src. Proc.	TCODE (000011)

Max length = 18 bits; Min length = 11 bits

Note: When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

70.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

70.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

70.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false. For the **evsel** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 1399. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**(1)
evsel instruction	00,01,10, or 11	evsel

1. If the EVCODE for direct branch function calls is not masked in DC4, taken bl and bcl instructions will generate Program Correlation Messages and will not be logged in the history buffer.

70.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

70.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow; or, if program trace is in history mode, since the last instruction that recorded history information.

Figure 1435. Resource Full Message format

(1–32 bits)	(4 bits)	(4 bits)	(6 bits)
RDATA	RCODE	Src. Proc.	TCODE (011011)

Max length = 46 bits; Min length = 15 bits

Table 1400 shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 1400. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

70.12.5 Program Correlation Messages

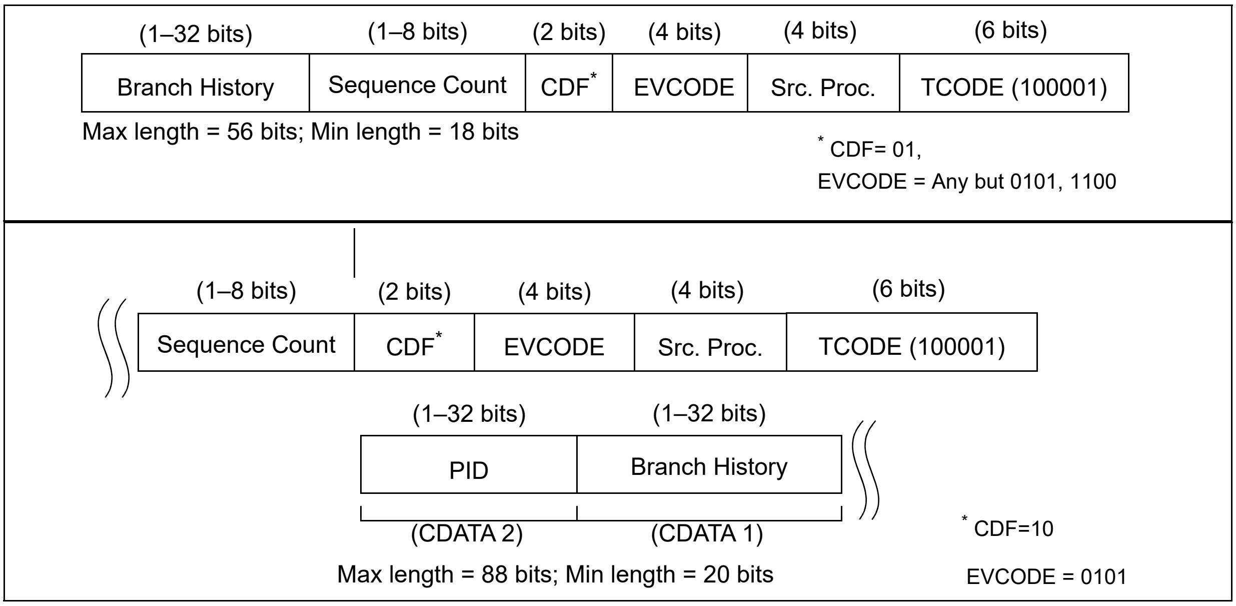
Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a “Branch and Link” instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}=‘0’ and $DC4_{PTMARK}$=‘1’.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to *Table 1372* for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

Figure 1436. Program Correlation message formats



70.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

70.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

70.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see *Table 1401*):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to MSR_{PMM} → ‘1’ with $DC4_{PTMARK}$=‘1’.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

Figure 1437. .Program Trace Synchronization message format

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Exception conditions that result in Program Trace Synchronization are summarized in *Table 1401*.

Table 1401. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

70.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see *Table 1401*):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync messages is as follows:

Figure 1438. Direct/Indirect Branch with Sync message format

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

The format for Program Trace Indirect Branch History with Sync. Messages is as follows:

Figure 1439. Indirect Branch History w/ Sync message format

(1–32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in *Table 1402*.

Table 1402. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG}→OTM → BTM → DTM)→ Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

70.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of four ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

70.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

Figure 1440. Program Trace — Indirect Branch message (Traditional)

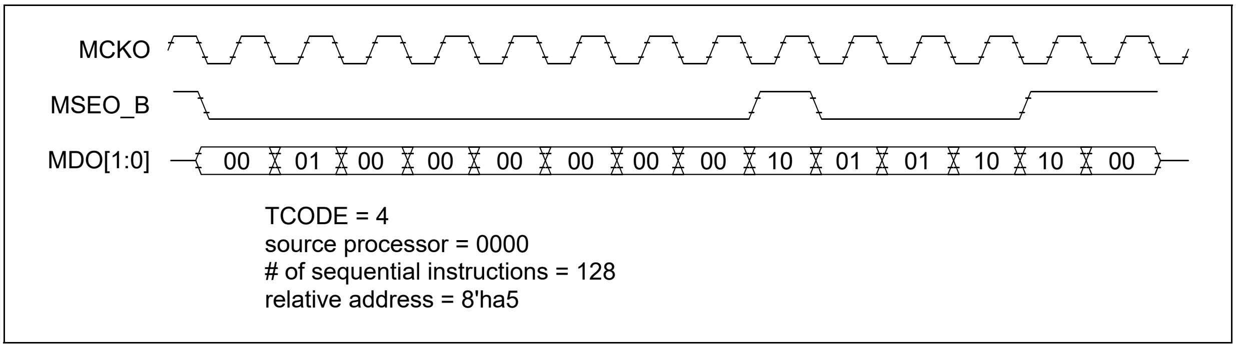


Figure 1441. Program Trace — Indirect Branch message (history)

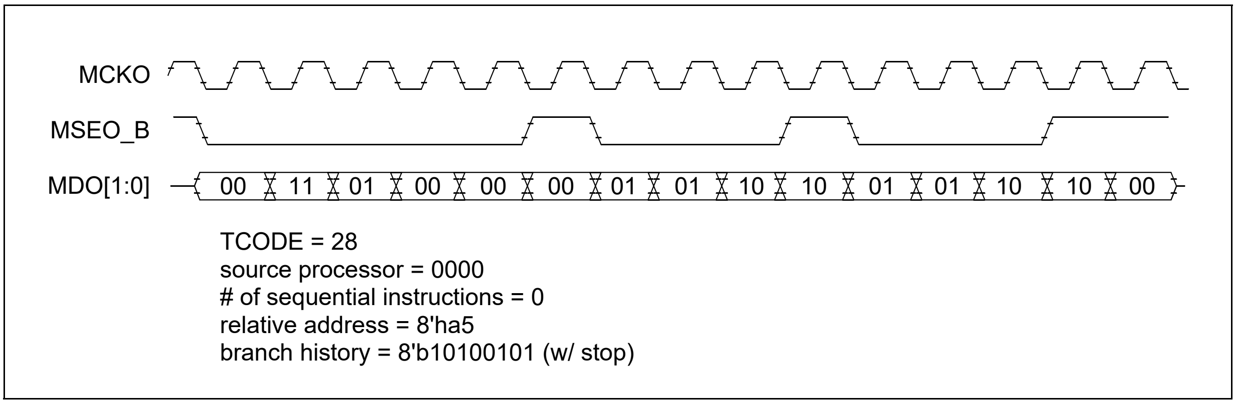


Figure 1442. Program Trace — Direct Branch (traditional) & Error messages

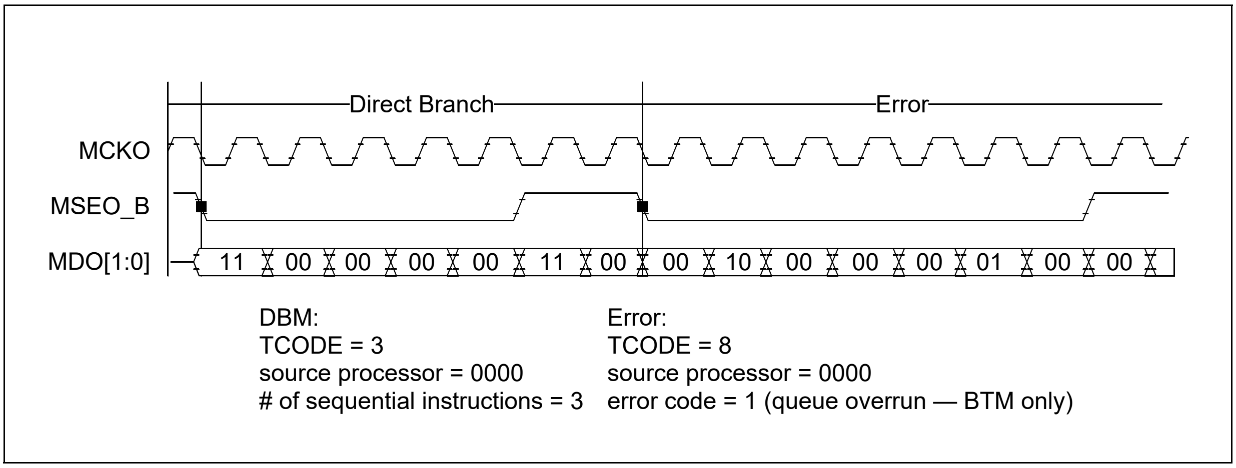
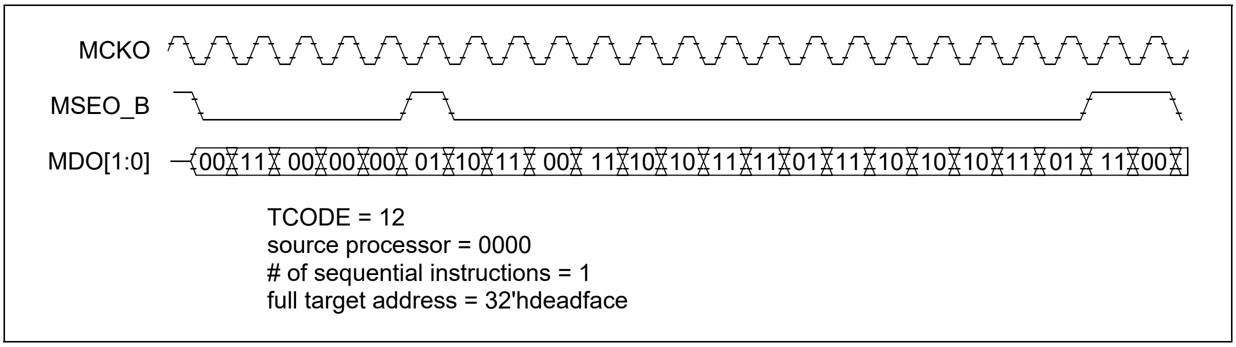


Figure 1443. Program Trace — Indirect Branch w/ Sync message



70.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

70.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note: Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z420n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**)

70.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization, and Error Messages.

70.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace message. Data Write message information is messaged out in the following format:

Figure 1444. Data Write message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

70.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

Figure 1445. Data Read message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

*Note: Core (e200z420n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.*

*For Core (e200z420n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.*

The debug/development tool will need to distinguish the two cases based on the family of processor.

70.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see *Table 1403*):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

Figure 1446. Data Write/Read with Sync message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

Exception conditions that result in Data Trace Synchronization are summarized in *Table 1403*.

Table 1403. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.

Table 1403. Data Trace Exception summary (continued)

Exception Condition	Exception Handling
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM)→ Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

70.13.3 DTM operation

70.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

70.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

70.13.3.3 Data Trace filtering

Data Trace filtering is available based on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

70.13.3.4 Bus cycle special cases

Table 1404. Bus cycle special cases

Special case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$)(1)	Data Trace Message discarded
Bus cycle completed without error(1)	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored

Table 1404. Bus cycle special cases (continued)

Special case	Action
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) — both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note: For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (that is word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z420n3) Core Debug Support chapter.)

Note: A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

70.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

Figure 1447. Data Trace — Data Write message

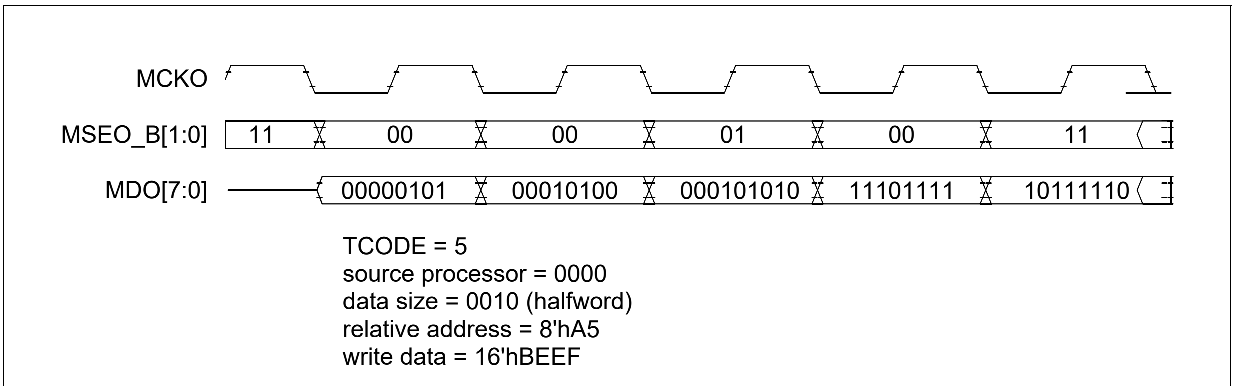
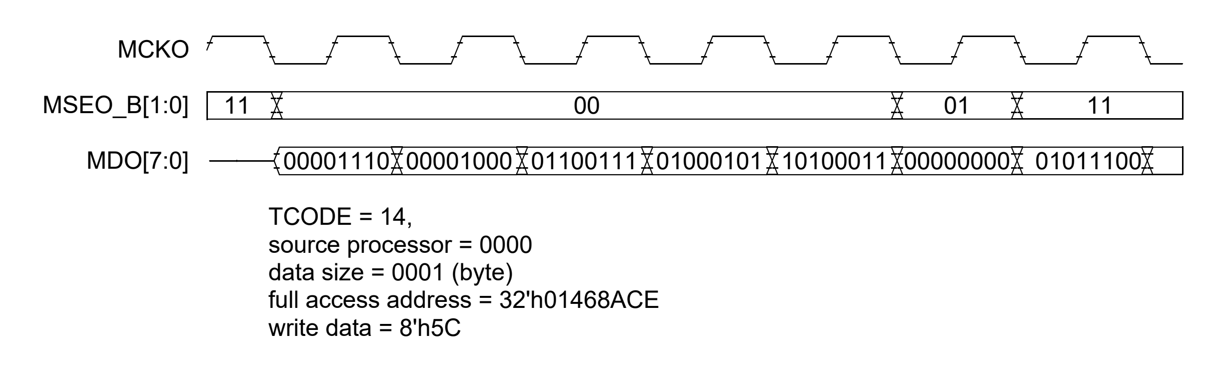


Figure 1448. Data Trace — Data Read w/ Sync message



70.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

70.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

70.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

70.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the “External Debug Resource Allocation Control Register (EDBRAC0)” section in the Core (e200z420n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

Table 1405. .Data Acquisition message format

(1–32 bits)	(8 bits)	(4 bits)	(6 bits)
DQDATA	DQTAG	Src. Proc.	TCODE (011011)

Max length = 50 bits; Min length = 19 bits

70.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z420n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See *Table 1409* for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

Figure 1449. Watchpoint message format

(1–30 bits)	(4 bits)	(6 bits)
Watchpoint Source	Src. Proc.	TCODE (001111)

Variable length = 11–40 bits

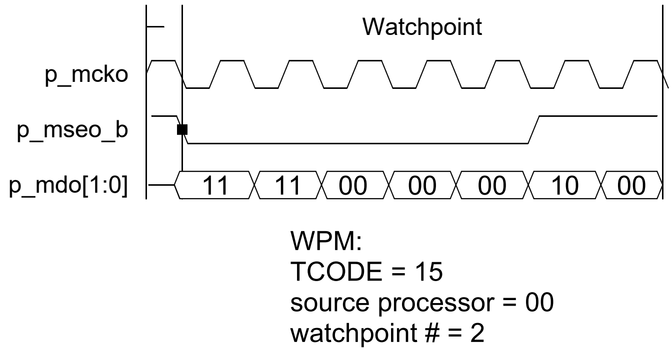
The Watchpoint Source message field will contain a ‘1’ for each asserted watchpoint. Leading zeros are truncated.

Table 1406. Watchpoint source encoding

Watchpoint source (1–30 bits)	Watchpoint description
000000000000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1	Watchpoint #0 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxx1x	Watchpoint #1 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxx1xx	Watchpoint #2 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxx1xxx	Watchpoint #3 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxx1xxxx	Watchpoint #4 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxx1xxxxx	Watchpoint #5 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxx1xxxxxx	Watchpoint #6 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #7 enabled for WTM
xxxxxxxxxxxxxxxxxxxxx1xxxxxxxx	Watchpoint #8 enabled for WTM
xxxxxxxxxxxxxxxxxxxx1xxxxxxxxx	Watchpoint #9 enabled for WTM
xxxxxxxxxxxxxxxxxxx1xxxxxxxxxx	Watchpoint #10 enabled for WTM
xxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #11 enabled for WTM
xxxxxxxxxxxxxxxxx1xxxxxxxxxxxx	Watchpoint #12 enabled for WTM
xxxxxxxxxxxxxxxx1xxxxxxxxxxxxx	Watchpoint #13 enabled for WTM
xxxxxxxxxxxxxxx1xxxxxxxxxxxxxx	Watchpoint #14 enabled for WTM
xxxxxxxxxxxxxx1xxxxxxxxxxxxxxx	Watchpoint #15 enabled for WTM
xxxxxxxxxxxxx1xxxxxxxxxxxxxxxx	Watchpoint #16 enabled for WTM
xxxxxxxxxxxx1xxxxxxxxxxxxxxxxx	Watchpoint #17 enabled for WTM
xxxxxxxxxxx1xxxxxxxxxxxxxxxxxx	Watchpoint #18 enabled for WTM
xxxxxxxxxx1xxxxxxxxxxxxxxxxxxx	Watchpoint #19 enabled for WTM
xxxxxxxxx1xxxxxxxxxxxxxxxxxxxx	Watchpoint #20 enabled for WTM
xxxxxxxx1xxxxxxxxxxxxxxxxxxxxx	Watchpoint #21 enabled for WTM
xxxxxxx1xxxxxxxxxxxxxxxxxxxxxx	Watchpoint #22 enabled for WTM
xxxxxx1xxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #23 enabled for WTM
xxxxx1xxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #24 enabled for WTM
xxxx1xxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #25 enabled for WTM
xxx1xxxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #26 enabled for WTM
xx1xxxxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #27 enabled for WTM
x1xxxxxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #28 enabled for WTM
1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #29 enabled for WTM

70.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

Figure 1450. Watchpoint Message & Watchpoint Error message



70.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a “window” for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect. *Table 1407* details the external hardware trigger signals.

Table 1407. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

Table 1407. External Hardware Trigger Signals (continued)

Name	Description
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0→1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

70.17 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6‘b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note: Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are <u>NOT</u> supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

70.17.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 70.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 70.5: Nexus 3 Register Access via JTAG/OnCE*. Configure the bits as follows:
   a) Access Control (AC) → 1b'1 (to indicate start access)
   b) Map Select (MAP) → 3b'000 (primary memory map)
   c) Access Priority (PR) → 2b'11 (highest priority)
   d) Read/Write (RW) → 1b'1 (write access)
   e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   f) Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in *Section 70.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see *Table 1409* for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note: Only the **nex_ahb_start, nex_rdy_b, and nex_err_b** *pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

70.17.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in *Section 70.17.1: Single write access* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has

been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.

3. If AC has not been cleared due to an error, repeat Step 3 in *Section 70.17.1: Single write access* until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

70.17.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 70.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 70.5: Nexus 3 Register Access via JTAG/OnCE*. Configure the bits as follows:
   a) Access Control (AC) → 1b'1 (to indicate start access)
   b) Map Select (MAP) → 3b'000 (primary memory map)
   c) Access Priority (PR) → 2b'11 (highest priority)
   d) Read/Write (RW) → 1b'0 (read access)
   e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   f) Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see *Table 1409* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 70.5: Nexus 3 Register Access via JTAG/OnCE*.

*Note: Only the **nex_ahb_start, nex_rdy_b,** and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

70.17.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in *Section 70.17.3: Single read access* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and

asserts the **nex_rdy_b** pin (see *Table 1409* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.

3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS Register, and asserts the **nex_rdy_b** pin (see *Table 1409* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 70.5: Nexus 3 Register Access via JTAG/OnCE*.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in *Section 70.17.3: Single read access* until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note: The data values must be shifted out 32 bits at a time LSB first (that is doubleword read = two word reads from the RWD).

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

70.17.5 Error handling

The Nexus 3 module handles various error conditions as follows:

70.17.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared).
2. The ERR bit in the RWCS Register is set.
3. The Error Message is sent (TCODE = 8) indicating Read/Write error.

70.17.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a) If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b) If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

70.17.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

Figure 1451. Error message format

(5 bits)	(4 bits)	(6 bits)
Error Code (00011)	Src. Proc.	TCODE (001000)

Fixed length = 15 bits

70.18 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

70.18.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in *Table 1409*. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see *Section 70.5: Nexus 3 Register Access via JTAG/OnCE* for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the “JTAG/OnCE Pins” section of the Core (e200z420n3) Core Debug Support chapter.

Table 1408. JTAG pins for Nexus 3

JTAG pins	Input/ Output	Description of JTAG pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the “Shift-IR” and “Shift-DR” controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in *Table 1409*.

Table 1409. Nexus 3 auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted

Table 1409. Nexus 3 auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_evto_b	O	Event Out (**nex_evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **nex_evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1) One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2) Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1) Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2) Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (**nex_wevto[3:0]**) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. **nex_wevto[3:0]** is held asserted for one (1) cycle of **nex_mcko**.
nex_ext_src_id[0:3]	I	nex_ext_src_id[0:3] is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time
nex_sfwcntl_en]	I	nex_sfwcntl_en is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to *Section 70.20: Auxiliary port arbitration*, for detail on Nexus port arbitration.

Table 1410. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See *Table 1412*.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant (**nar_aux_grant**) is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

70.18.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in *Table 1411*. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

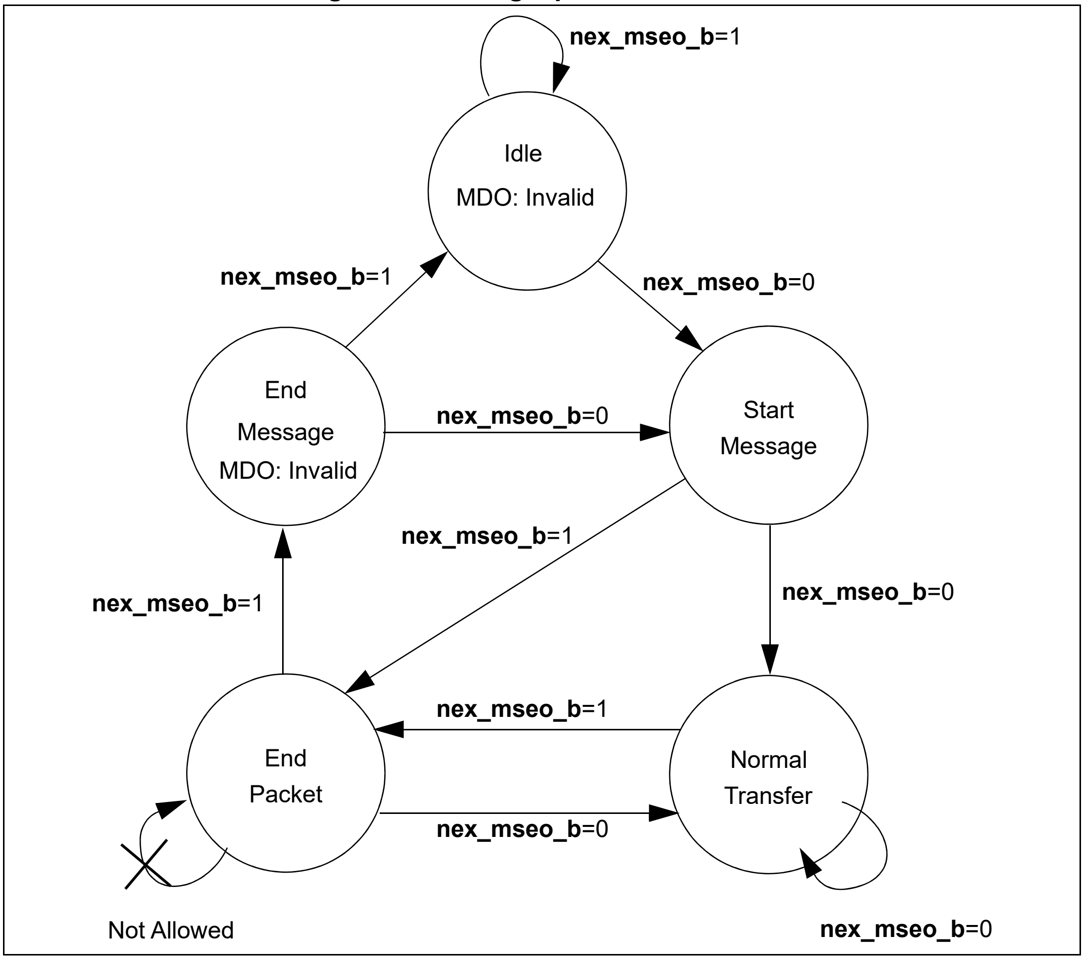
Single pin MSEO is not supported on the SPC584Bx.

Table 1411. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1’s)	00 (or 01)-11-(more 1’s)
End of variable length packet	0-1-0	00-01
Message transmission	0’s	00’s
Idle (no message)	1’s	11’s

Figure 1452 illustrates the state diagram for single pin MSEO transfers.

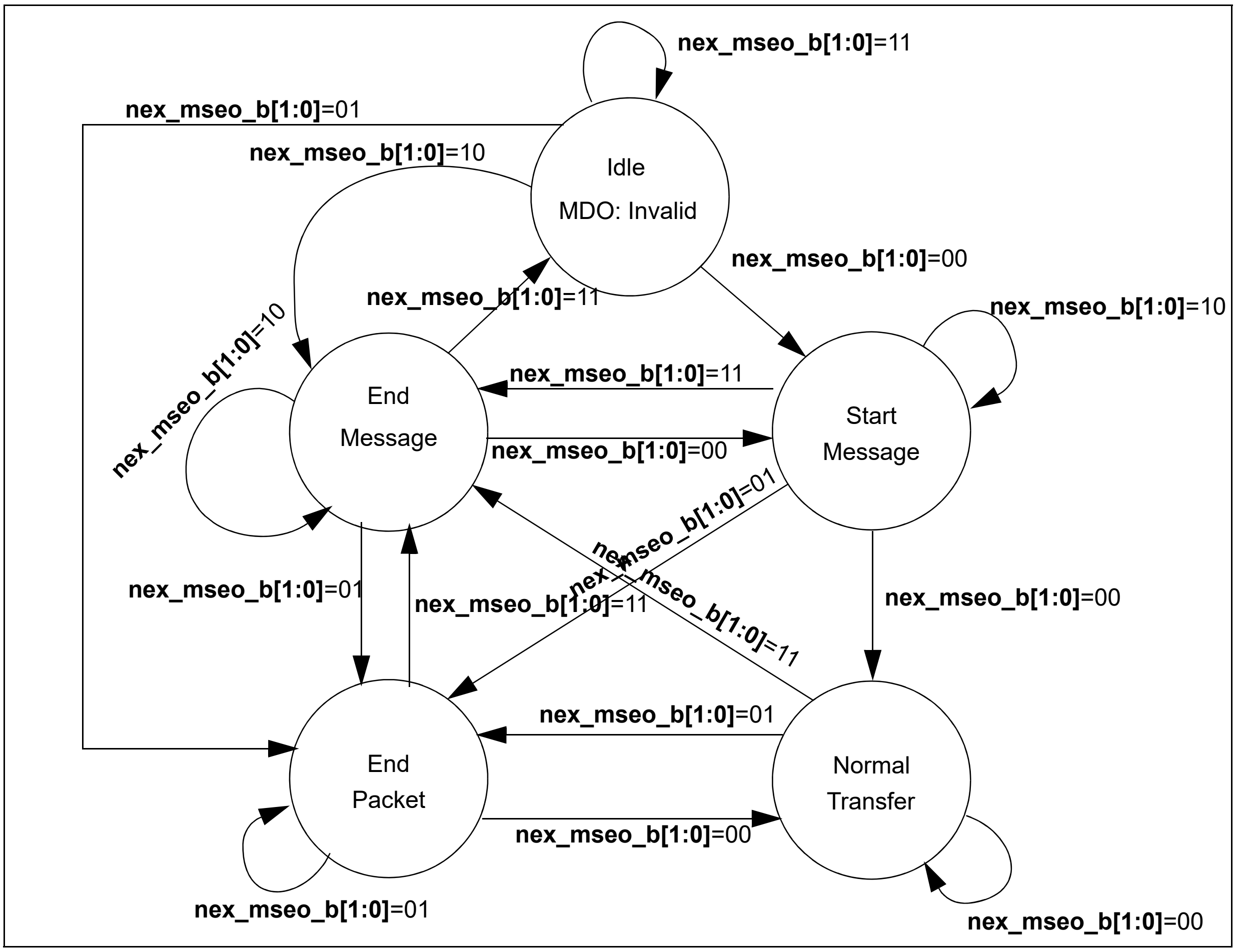
Figure 1452. Single pin MSE transfers



Note that the “End Message” state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive “End Packet” messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive ‘1’s on the **nex_mseo_b** pin before the actual end of message.

Figure 1453 illustrates the state diagram for dual pin MSEO transfers.

Figure 1453. Dual pin MSEO transfers



The dual pin MSEO option is more robust than the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive “End Packet” states. This can be an advantage when small, variable sized packets are transferred.

Note: The “End Message” state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

70.19 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-

order bit so that it can end on a port boundary.

For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

70.20 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in *Table 1412* below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in *Section 70.18.2: Pin protocol*. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 1412. MDO request encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

70.21 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 1413 illustrates an example of Indirect Branch Message with 2 MDO / 1MSEO configuration. *Table 1414* illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 1413. Indirect branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 1414. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	MAP	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 1415 and *Table 1416* illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 1415. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 1416. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 1417 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and *Table 1418* illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 1417. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 1418. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

70.22 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the SPC584Bx Data Sheet.

70.23 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

70.23.1 JTAG sequence for accessing internal Nexus registers

Table 1419. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE → SELECT-DR_SCAN
2	0	SELECT-DR_SCAN → CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR → SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR → EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR → UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR → SELECT-DR_SCAN
8	0	SELECT-DR_SCAN → CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR → SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR → EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR → UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR → RUN-TEST/IDLE (transfer complete — Nexus controller to Reg. Select state)

70.23.2 JTAG sequence for read access of memory-mapped resources

Table 1420. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

70.23.3 JTAG sequence for write access of memory-mapped resources

Table 1421. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value — data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address — data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
8	—	If CNT > 0, go back to Step #5

71 Nexus Crossbar Multi-Master Client (NXMC)

71.1 Introduction

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2012 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters such as DMA AHB, and CPU Data Bus AHB start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools.This also provides the capability of sending user defined messages for the peripherals attached to it.

71.1.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads or writes, or both, to (selected) internal memory resources
- Multi-master tracing capability for multiple master accesses
- HSM master(s) filtering from the tracing function based on external control
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging
- Peripheral interface for data message transfers based on In Circuit Trace Message (ICTM) format
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on external triggers
- Optional timestamping of the messages based on timer value and controls received at the interface
- Nexus Auxiliary port for higher data throughput
  - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
  - One $\overline{\text{EVTI}}$ (Event In) pin
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

71.2 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NPC. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$

These pins are described in *Table 1423*.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (refer to *Section 71.5.1: IEEE 1149.1 (JTAG) Test Access Port* for details). A separate JTAG TAP controller is instantiated within the NXMC. *Table 1422* describes the JTAG pins.

Table 1422. JTAG pins for NXMC

JTAG port	Input/ output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 1423. NXMC auxiliary pins

Auxiliary port	Input/ output	Description
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: – one of the internal or external watchpoints has occurred and EOC = 2'b00 – system-level debug mode was entered (ipg_debug) and EOC = 2'b01 $\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (refer to *Section 71.5.2: Enabling NXMC operation*), assertion initiates data trace synchronization messages (provided data trace is enabled).

71.3 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number or size of the packets, or both, to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in *Table 1424*. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 1424. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (refer to *Table 1425*)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (refer to *Table 1425*)
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (refer to *Table 1426*)
	14	14	ECODE	fixed	Error code (refer to *Table 1427*)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Write Message with Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (refer to *Table 1425*)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 1424. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Read Message with Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (refer to *Table 1425*)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (refer to *Table 1439*)
	1	26	TSTAMP	variable	Timestamp (Optional)
Peripheral Message (In circuit Trace Message: ICTM)	6	6	TCODE	fixed	TCODE number = 34
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	6	6	CKSRC	fixed	Peripheral source identifier
	1	44	CKDATA	variable	Peripheral data
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 1425. Data trace size (DSZ) encodings (TCODE = 5,6,13,14)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 1426. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved

Table 1426. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 1427. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description(1)
13 to 8	7 to 0	
Source (SRC)		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	In-Circuit Trace Message(s) Lost
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

71.4 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. Refer to *Section 71.5.1: IEEE 1149.1 (JTAG) Test Access Port* for details. The complete list of registers is shown in *Table 1428*.

Table 1428. NXMC registers

Nexus register	Nexus access opcode	Read/ write	Read address	Write address
Development Control 1 (DC1)	0x2	R/W	0x04	0x05
Development Control 2 (DC2)	0x3	R/W	0x06	0x07
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B
Data Trace Start Address1 (DTSA1)	0xE	R/W	0x1C	0x1D
Data Trace Start Address2 (DTSA2)	0xF	R/W	0x1E	0x1F
Data Trace End Address1 (DTEA1)	0x12	R/W	0x24	0x25
Data Trace End Address2 (DTEA2)	0x13	R/W	0x26	0x27
Breakpoint/Watchpoint Control Register1 (BWC1)	0x16	R/W	0x2C	0x2D
Breakpoint/Watchpoint Control Register2 (BWC2)	0x17	R/W	0x2E	0x2F
Breakpoint/Watchpoint Address Register1 (BWA1)	0x1E	R/W	0x3C	0x3D
Breakpoint/Watchpoint Address Register2 (BWA2)	0x1F	R/W	0x3E	0x3F

71.4.1 Register definitions

Detailed register definitions for the current NXMC implementation are described in the following sections.

71.4.1.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. *Figure 1454* shows the format of the DC1.

Address: refer to *Table 1428* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	EOC		0	0	WEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	EIC		TM		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not implemented and its value has no impact on the system.

Figure 1454. Development control register 1 (DC1)

The DC1 is described in *Table 1429*.

Table 1429. DC1 register field description

Field	Description
3:4 EOC	$\overline{\text{EVTO}}$ Control. 00 $\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal or external) 01 $\overline{\text{EVTO}}$ upon entry into system-level Debug Mode 1x Reserved
7 WEN	Watchpoint Trace Enable. 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
27:28 EIC	$\overline{\text{EVTI}}$ Control 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled 11 Reserved
29:31 TM	Trace Mode 000 No Trace 1xx Reserved x1x Data Trace enabled xx1 Reserved

71.4.1.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. *Figure 1455* shows the format of the DC2.

Address: refer to *Table 1428* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PME0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not implemented and its value has no impact on the system.

Figure 1455. Development control register 2 (DC2)

The DC2 is described in *Table 1430*.

Table 1430. DC2 register field description

Field	Description
0:7 EWC	$\overline{\text{EVTO}}$ Watchpoint Configuration[1]. 00000000 No Watchpoints trigger $\overline{\text{EVTO}}$ 1XXXXXXX External Watchpoint #1 triggers $\overline{\text{EVTO}}$ X1XXXXXX External Watchpoint #2 triggers $\overline{\text{EVTO}}$ XX1XXXXX Reserved[2] XXX1XXXX Reserved[2] XXXX1XXX Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$ XXXXX1XX Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$ XXXXXX1X Reserved[2] XXXXXXX1 Reserved[2]
31 PME0	Peripheral Messaging Enable for the In-Circuit Trace (ICT) feature.[3] 0 Peripheral Interface disabled 1 Peripheral Interface enabled

1. The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.
2. Writing bits 2,3,6,7 has no impact and read value will be 0 for these bits.
3. The peripheral details are described in the device-specific chapter that describes how the modules are configured.

71.4.1.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable. The WT bits can be used to produce an address related window for triggering trace messages. *Figure 1456* shows the format of the WT register.

Address: refer to *Table 1428* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	DTS			DTE			0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1456. Watchpoint trigger register (WT)

The WT register is described in *Table 1431*.

Table 1431. WT register field description

Field	Description[1]
6;8 DTS	Data Trace Start Control. 000 Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
9;11 DTE	Data Trace End Control. 000Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved

1. The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

71.4.1.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXMC module. *Figure 1457* shows the format of the DTC register.

Address: refer to *Table 1428* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RWT1		RWT2		0	0	0	MME	0	0			MID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1457. Data trace control (DTC) register

The DTC register is described in *Table 1432*.

Table 1432. DTC field description

Field	Description
0:1 RWT1	Read/Write Trace 1. 00 No trace messages generated x1 Enable data read trace 1x nable data write trace
2:3 RWT2	Read/Write Trace 2. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
7 MME	Multi Master Tracing enable. 0 Tracing enabled only for Master with ID = MID (DTC[19:16]) 1 Tracing enabled for all the Masters
12:15 MID	ID of the system bus Master to be traced(1)
24 RC1	Range Control 1. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
25 RC2	Range Control 2. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)

1. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

71.4.1.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. *Figure 1458* shows the format of the DTSA1 and DTSA2 registers.

Address: refer to *Table 1428* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA TRACE START ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA TRACE START ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1458. Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

71.4.1.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. *Figure 1459* shows the format of the DTEA1 and DTEA2 registers.

Address: refer to Table 1428 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	DATA TRACE END ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	DATA TRACE END ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1459. Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

Table 1433 illustrates the ranges that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 1433. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range – no trace

Note: DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.

When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

71.4.1.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. Figure 1460 shows the format of the BWC1 register.

Address: refer to Table 1428 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1460. Breakpoint/watchpoint control register 1(BWC1)

The BWC1 register is described in Table 1434.

Table 1434. BWC1 field description

Field	Description
0:1 BWE1	Breakpoint/Watchpoint #1 Enable 00 Internal Nexus Watchpoint #1 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #1 enabled
2:3 BRW1	Breakpoint/Watchpoint #1 Read/Write Select 00 Watchpoint #1 hit on read accesses 01 Watchpoint #1 hit on write accesses 10 Watchpoint #1 on read or write accesses 11 Reserved
14:15 BWR1	Breakpoint/Watchpoint #1 Register Compare 00 No Register Compare (same as BWC1[31:30] = 2'b00) 01 Reserved 10 Compare with BWA1 value 11 Reserved
16 BWT1	Breakpoint/Watchpoint #1 Type 0 Reserved 1 Watchpoint #1 on data accesses

71.4.1.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. *Figure 1461* shows the format of the BWC2 register.

Address: refer to *Table 1428* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1461. Breakpoint/watchpoint control register 2(BWC2)

The BWC2 register is described in *Table 1435*.

Table 1435. BWC2 field description

Field	Description
0:1 BWE2	Breakpoint/Watchpoint #2 Enable 00 Internal Nexus Watchpoint #2 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #2 enabled
2:3 BRW2	Breakpoint/Watchpoint #2 Read/Write Select 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
14:15 BWR2	Breakpoint/Watchpoint #2 Register Compare. 00 No Register Compare (same as BWC2[31:30] = 2b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved
16 BWT2	Breakpoint/Watchpoint #2 Type 0 Reserved 1 Watchpoint #2 on data accesses

71.4.1.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. *Figure 1462* shows the format of the BWA1 and BWA2 registers.

Address: refer to *Table 1428* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BREAKPOINT / WATCHPOINT ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BREAKPOINT / WATCHPOINT ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1462. Breakpoint /watchpoint address registers 1 and 2 (BWA1 and BWA2)

71.4.1.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in *Table 1428*. For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

71.5 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of $\overline{\text{TRST}}$ and system reset de-assertion, after the JTAG ID register has been read by the tool after negation of $\overline{\text{TRST}}$ and should be programmed before deassertion of system reset.

71.5.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in *Section 71.5.4: NXMC register access via JTAG*.

71.5.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

71.5.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set.

71.5.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in *Table 1428*.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (refer to *Table 1428*), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown in *Figure 1463* and *Table 1436*.

	7	6	5	4	3	2	1	0
R W	Nexus Register Index							R/W
Reset	0	0	0	0	0	0	0	0

Figure 1463. JTAG DR for Nexus register access

Table 1436. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in *Table 1428*
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

71.6 Functional description

71.6.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2012 standard.

71.6.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

71.6.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

71.6.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32, 64. The TSTAMP field is 26 bits. Data write message and data read message information is messaged out in the format shown in *Figure 1464*. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

8/16/32/64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010 or 111011)

Figure 1464. DTM message format for DATA width = 64 bits

TSTAMP is another optional field which is inserted based on the controls received from the NPC, refer to *Section 71.6.4: Timestamping feature*.

71.6.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

71.6.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in *Figure 1465*. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

Figure 1465. DTM error message format (fixed length = 48 bits)

71.6.1.2.3.1 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (refer to *Table 1437*):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message
- Upon returning from a low power state, the first data trace message is a synchronization message
- Upon returning from debug mode, the first data trace message is a synchronization message
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred
- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message.This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in *Figure 1466*. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

8-64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100 or 111101)

Figure 1466. Data write/read with synchronization message format

Exception conditions that result in data trace synchronization are summarized in *Table 1437*.

Table 1437. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.

Table 1437. Data trace exception summary (continued)

Exception condition	Exception handling
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).
Collision Priority	All messages have the following priority: Error > WPM > ICTM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

71.6.1.3 DTM operation

71.6.1.3.1 Enabling data trace messaging

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).
- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note: When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (that is DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure the correct DTM functionality.

71.6.1.3.2 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

Note: If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > ICTM > DTM).

71.6.1.3.3 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2012 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address (A1) = 0x0003FC01, the new address (A2) = 0x0003F365 as shown in *Figure 1467*.

Figure 1467. Relative addressing example

Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 1111 0110 0100

Address Message (M1) = 1111 0110 0100

Address Re-creation:

A1 ⊕ M1 = A2
A1 = 0000 0000 0000 0011 1111 1100 0000 0001
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

71.6.1.3.4 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

71.6.1.3.5 System bus cycle special cases

The system bus cycle special cases are shown in *Table 1438*.

Table 1438. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

71.6.1.3.6 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (refer to figure in *Section 71.6.1.2.1: Data write and data read messages* for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

71.6.1.3.7 HSM master filtering

For security reasons certain HSM masters need to be disabled or filtered from data tracing operations.

71.6.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2012. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

71.6.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note: In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (that are BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints: Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in *Section 71.6.1.3.6: System bus multi-master tracing*).
- External watchpoints: The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level.The watchpoints are described in the device-specific chapter that describes how the modules are configured. If either of these signals asserts (that is on 0-to-1 transition), a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in *Figure 1468* and *Table 1439*.

8 bits	4 bits	6 bits
WPHIT (RRRRXXXX)	SRC	TCODE (001111)

Figure 1468. Watchpoint message format (fixed length = 18 bits)

Table 1439. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1(1)	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

1. R = reserved for future use (insert 0s).

71.6.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in *Figure 1469*.

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

Figure 1469. Error message format (fixed length = 28 bits)

71.6.3 Peripheral interface

The NXMC provides an asynchronous interface to peripherals for sending data messages. The NXMC can back pressure or delay the data transfer using a data accept signal. A round-robin scheme is followed between the peripherals for forwarding received data to the queue; therefore, each peripheral has an equal priority and hence bandwidth for data transfers.

To enable messaging from peripherals there are controls present in the DC2 register for individual peripherals: DC2[PME0]. There are no message generated and hence data accepted from a peripheral unless the corresponding enable bit is set to 1.

MSB	6 bits		(54-6) bits				LSB
	CKSRCn		DATAn				
54-1		54-6	54-6-1		2	1	0

n = peripheral number

Figure 1470. Peripheral DATA bus fields

Another input port is present which indicates data loss inside the peripheral itself due to over-run, the module in turn generates error message to communicate such events to the debugger.

71.6.3.1 Peripheral data message format (ICTM)

The message formats for the peripheral data messages (formatted as ICTM) is shown in *Figure 1471*. The SRC field values are described in the device-specific chapter that describes how the modules are configured. The CKSRC field is received from the MSB 6 bits of the individual peripheral data bus, and CKDATA is a variable field and contains 44 LSBits received on the databus of the peripheral interface.

1–44 bits	6 bits	4 bits	6 bits
CKDATAn	CKSRCn	SRC	TCODE (100010)

Figure 1471. Peripheral message format (max length = 60 bits, min length = 17 bits)

71.6.3.2 Peripheral error message format

The error messages are generated based data over-run indication from a peripheral. Error information is messaged out in the format shown in *Figure 1472*.

14 bits	4 bits	4 bits	6 bits
ECODE(CKSRCn,ErrCODE)	ETYPE	SRC	TCODE (001000)

Figure 1472. Error message format (fixed length = 28 bits)

The SRC field values are described in the device-specific chapter that describes how the modules are configured. ETYPE and ECODE encoding are listed in *Table 1426* and *Table 1427*.

71.6.4 Timestamping feature

This feature allows a timestamp packet to be appended to all messages, based on the timer value and controls received externally from the NPC. If enabled, this packet is added as the last field to all the outgoing messages. The TSTAMP packet width is 26 bits.

The TSTAMP packet is appended at the end of a message during message formatting, before transfer. *Figure 1473* shows an example of ICTM message with a TSTAMP field.

26 bits	1–44 bits	6 bits	4 bits	6 bits
TSTAMP	CKDATAn	CKSRCn	SRC	TCODE (100010)

Figure 1473. Message with TSTAMP field example

The enable or disable indication should remain static for the complete debug session. Dynamically enabling or disabling of the timestamping is not allowed during running debug tracing. The following steps allow seamless enabling/disabling of the timestamping:

1. All tracing (DTM/WPM/ICTM) must be disabled
2. Wait for the internal queues to be emptied (no further request is asserted at the auxiliary interface)
3. Change the timestamp enable indication to assert/deassert
4. Enable the required tracing controls (via such as TM, PME0 control bits)

72 Cyclic Redundancy Check (CRC) Unit

72.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading this work from the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of four hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

72.2 Main features

- 4 contexts for the concurrent CRC computation
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- 4 hard-wired polynomials (two CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)
  - Support for byte, halfword, or word width of the input data stream

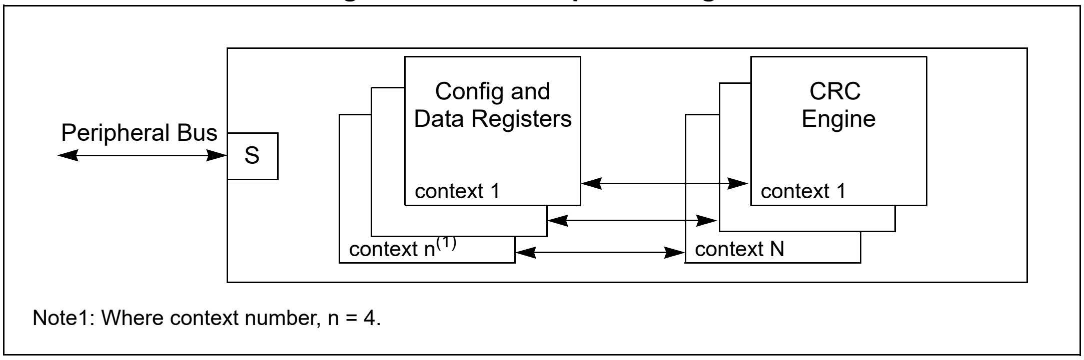
72.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT
- CRC-32 Ethernet
- CRC-8-H2F Autosar polynomial

72.3 Block diagram

The top level diagram of the CRC module is given in *Figure 1474*.

Figure 1474. CRC top level diagram



72.4 Signal description

The CRC module does not generate any external signals.

72.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers

Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are accessible (read/write) in each access mode: user, supervisor, and test.

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

72.5 Memory map and registers

This section describes, in address-order, all the CRC registers.

72.5.1 Register description

Table 1440 shows the memory map for the CRC registers.

Table 1440. CRC unit memory map

Offset	Register	Location
CRC registers per context (n=0 to CRC_CNTX_NUM - 1 with CRC_CNTX_NUM = 4)		
0x000 + *n**0x10	CRC Configuration Register *n* (CRC_CFG*n*)	Section 72.5.1.1
0x004 + *n**0x10	CRC Input Register *n* (CRC_INP*n*)	Section 72.5.1.2
0x008 + *n**0x10	CRC Current Status Register *n* (CRC_CSTAT*n*)	Section 72.5.1.3
0x00C + *n**0x10	CRC Output Register *n* (CRC_OUTP*n*)	Section 72.5.1.4

72.5.1.1 CRC Configuration Register *n* (CRC_CFG*n*)

Offset: 0x000 + *n* *0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	SWAP_BYTEWISE	SWAP_BITWISE	POLYG		SWAP	INV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	f(n)(1)	0	0

Figure 1475. CRC Configuration Register *n* (CRC_CFG*n*)

1. Reset value is a function of n. When n%2 = 0, reset value = 0. When n%2 = 1, reset value = 1.

Table 1441. CRC_CFG*n* field descriptions

Field	Description
26 SWAP_BYTEWISE	SWAP_BYTEWISE 0 No swap 1 Byte wise (MSB to LSB, LSB to MSB) bytes swap performed on CRC_INP input data internally for CRC-32 polynomial calculations **Note:** INV and SWAP bits are set to ‘1’ for CRC-32 polynomial calculations.
27 SWAP_BITWISE	SWAP_BITWISE 0 No swap 1 Bit wise (MSB to LSB, LSB to MSB) bits swap performed on CRC_INP input data internally for CRC-16 polynomial calculations

Table 1441. CRC_CFG*n* field descriptions (continued)

Field	Description
28:29 POLYG	Polynomial selection 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 CRC-8H2F Autosar polynomial This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.
30 SWAP	SWAP selection 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content. The swap operation is a bit-by-bit rotation of the content. In case of CRC-CCITT polynomial, the swap operation is applied on the 16 LSB bits. This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.
31 INV	INV selection 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content. The inversion operation is a complement or negation of the content. In case of CRC-CCITT polynomial the inversion operation is applied on the 16 LSB bits. This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.

72.5.1.2 CRC Input Register *n* (CRC_INP*n*)

Offset: 0x004 + *n* * 0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1476. CRC Input Register *n* (CRC_INP*n*)

Table 1442. CRC_INP*n* field descriptions

Field	Description
0:31 INP	Input data for the CRC computation The INP register can be written at byte, halfword (high and low), or word in any sequence. In case of halfword write operation, the bytes must be contiguous. Only the bits written are fed to the CRC engine. This register can be read and written by software.

72.5.1.3 CRC Current Status Register *n* (CRC_CSTAT*n*)

Offset: 0x008 + *n* * 0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	CSTAT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	CSTAT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1477. CRC Current Status Register *n* (CRC_CSTAT*n*)

Table 1443. CRC_CSTATn field descriptions

Field	Description
0:31 CSTAT	Status of the CRC signature The CSTAT register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the16 LSB bits are significative. The 16 MSB bits are tied at 0b during the computation. In the case of the CRC-8 polynomial, only the 8 LSB bits are significative. The 24 MSB bits are tied at 0b during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be read and written by software. **Note:** This register can be written only during the configuration phase.

72.5.1.4 CRC Output Register *n* (CRC_OUTP*n*)

Offset: 0x00C + *n**0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OUTP															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTP															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1478. CRC Output Register *n* (CRC_OUTP*n*)

Table 1444. CRC_OUTPn field descriptions

Field	Description
0:31 OUTP	Final CRC signature The OUTP register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the16 LSB bits are significative. The 16 MSB bits are tied at 0b during the computation. In the case of the CRC-8 polynomial, only the 8 LSB bits are significative. The 24 MSB bits are tied at 0b during the computation. This register can be read by software.

72.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less than or equal to the number of contexts.

- The standard generator polynomials are given in *Equation 50*, *Equation 51* and *Equation 52* for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

Equation 50 CRC8 VDA CAN:

$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 51 CRC-CCITT (x.25 protocol):

$$x^{16} + x^{12} + x^5 + 1$$

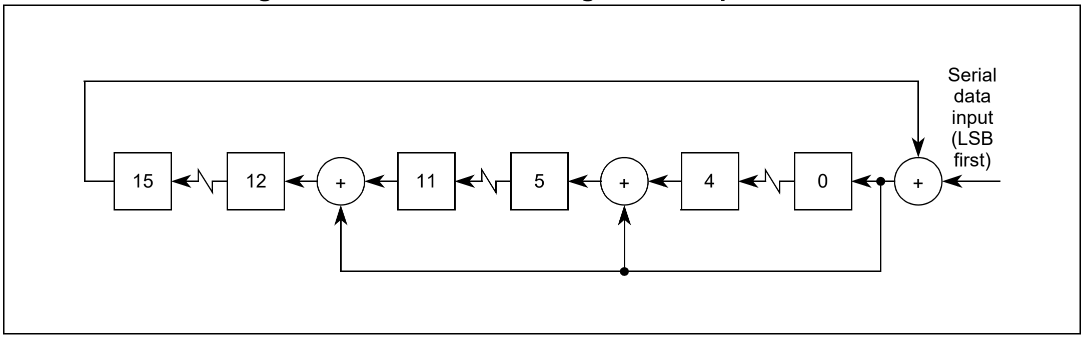
Equation 52 CRC-32 (Ethernet protocol):

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Equation 53 CRC-8-H2F (Autosar 4.0 protocol):

$$x^8 + x^5 + x^3 + x^2 + x + 1$$

Figure 1479. CRC-CCITT engine concept scheme

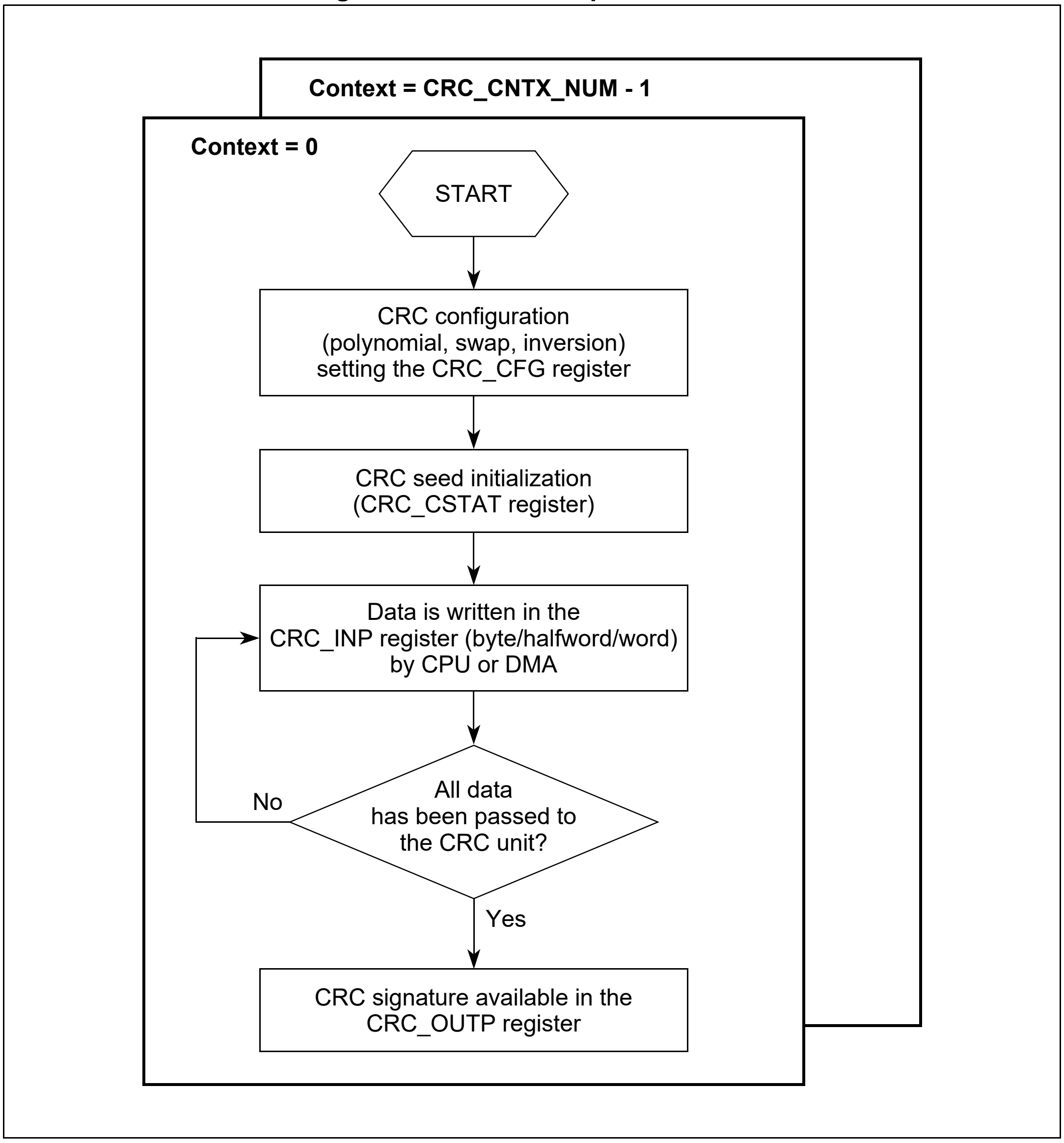


The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. The concept scheme (serial data loading) of the CRC engine is given in *Figure 1479* for the CRC-CCITT. The design implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the peripheral bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (such as SPI), a bit swap (MSB → LSB, LSB → MSB) or bit inversion of the final CRC signature, or both can be applied (CRC_OUTP register) before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in *Figure 1480*.

Figure 1480. CRC computation flow



72.7 Use cases

72.7.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

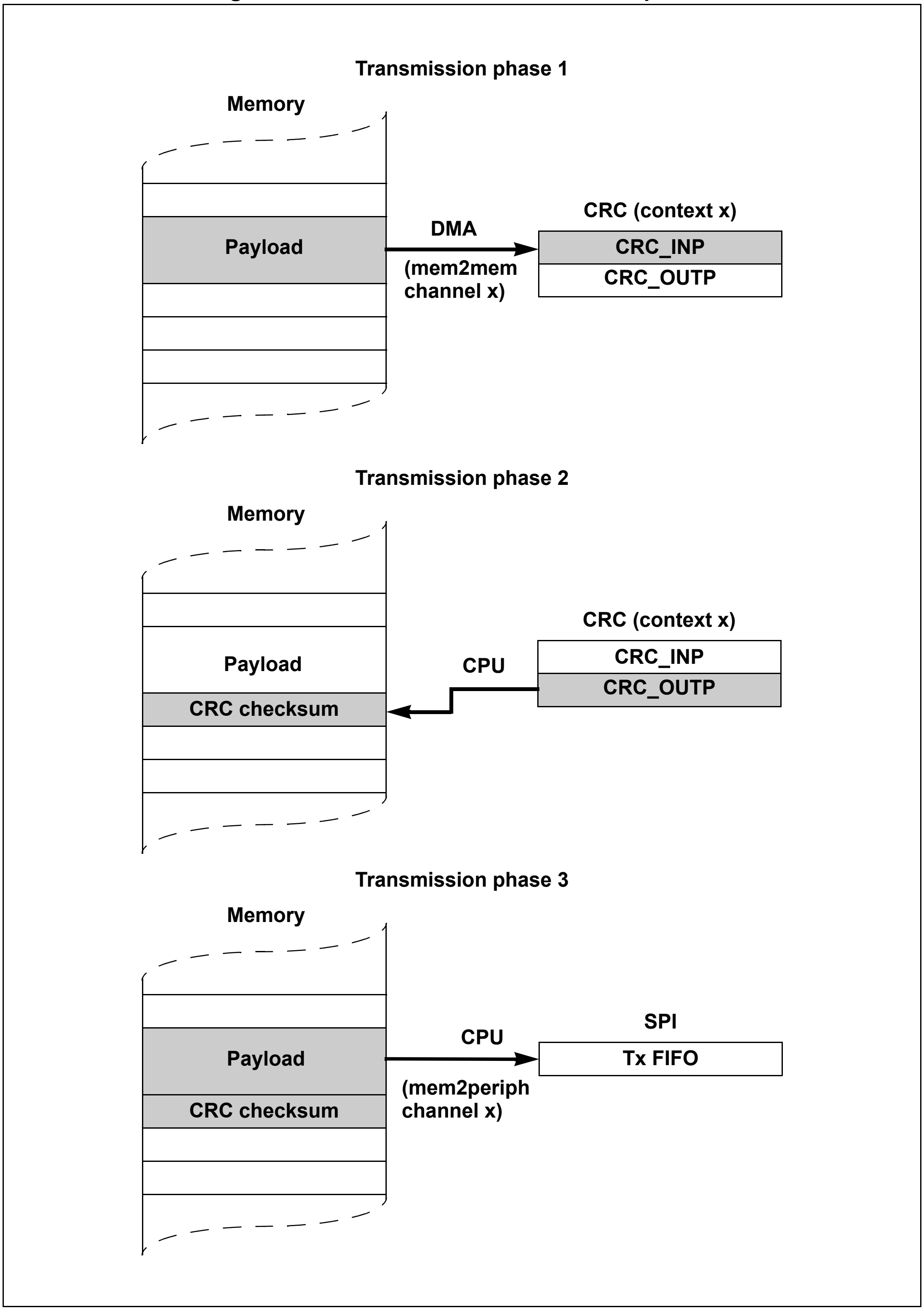
- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has *n* channels (greater than or equal to the number of contexts) configurable for the following types of data transfer: mem2mem, periph2mem, mem2periph; then this sequence, as shown in *Figure 1481*, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x, channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB → LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

Figure 1481. DMA-CRC transmission sequence



72.7.2 Register programming

- INV or SWAP, or both (affecting the output only) can be applied to CRC-8, CRC-16 and CRC-32. Both can be applied together.
- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the Ethernet standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

73 Memory Error Management Unit (MEMU)

73.1 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- SRAM
- Peripheral RAM
- Flash memory.

When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
  - Single/Double bit error in the data part that is detected via ECC for flash memory. Reporting of single/double bit error as correctable error depends on CTRL.FLASH_ERR_SELECT configuration.
  - Single Bit error in the data part that is detected via MBIST on any RAM.
- Uncorrectable Error: It comprises the following:
  - Double/Multiple (more than two) bit error that is detected via ECC for Flash Memory. Reporting of double bit error as uncorrectable error depends on CTRL.FLASH_ERR_SELECTION configuration.
  - Multiple bit error that is detected via MBIST on any SRAM. Based on MBIST implementation, the MBIST might report several correctable errors in the same word instead of an uncorrectable one. It is the task of the software after MBIST to aggregate these reports and detect the uncorrectable error.
  - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.

When multiple errors are indicated from various sources at the same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from either ECC or MBIST logic), independently of the source generating these errors.

The MEMU has multiple instances of the basic reporting block; one each for:

- System RAM ECC and MBIST
- Peripheral SRAM ECC
- Flash memory ECC.

Each instance has two reporting tables–Correctable errors are reported in one table while the uncorrectable are reported in the other.

See *Section 73.4: Design overview* for the reporting block details.

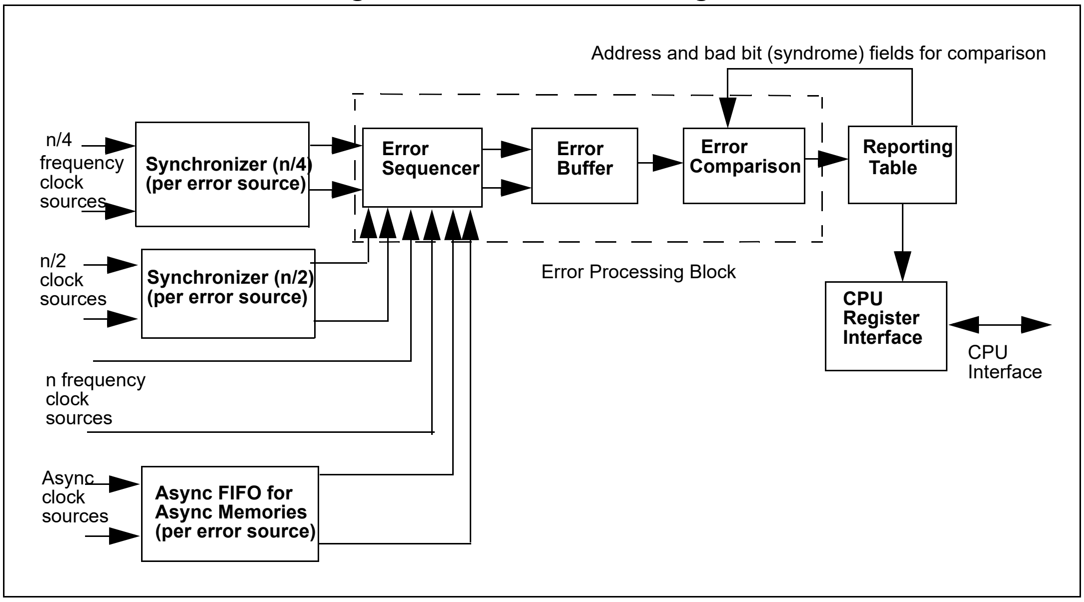
73.2 Features

The MEMU has the following features:

- Supports up to 128 error sources per Reporting Block (System RAM ECC and MBIST, Peripheral SRAM ECC or flash memory ECC): The number of error sources are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.
- Support for error reporting from the following category of error sources (both for ECC and MBIST):
  - System RAM
  - Peripheral SRAM
  - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.
- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.
- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

73.3 Block diagram

Figure 1482. MEMU block diagram



73.4 Design overview

The MEMU block has three instantiations of the basic reporting block along with a common CPU interface for the user software to access the storage block.

The MEMU should be enabled on power-on reset. The user software can write known error addresses into the reporting table to prevent reporting of those errors addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See *Section 73.7.3: Handling overflows (Multiple error reporting)* for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

   For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

   For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

2. Error Processing block

   This block is used to process the incoming errors coming from the device.

   – Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.

   – All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.

   – While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit/syndrome (depending on ECC or MBIST logic) of the incoming error with the one stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set, else the incoming error is dropped.

   – While doing the error buffer uniqueness check, the type of error (Correctable Error: CE or Uncorrectable Error: UCE) is also compared, hence even if the Error Address and the Bad bit/syndrome fields (depending on whether the incoming error is generated from ECC or MBIST logic) of the incoming error matches with the error buffer, if the type of errors (CE and UCE) differ they are treated as unique.

- Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable).
- Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (<=5 unless the error gets temporarily stored in the error buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

   This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

   The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

   The CPU can clear the flags signaling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (*System RAM correctable error reporting table status register (SYS_RAM_CERR_STSn)* or *System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)*). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

   The reporting table sizes are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.

4. Register block

   The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signaled to the CPU and the CPU command will be completed in the next clock cycle.

   The software, in principle, is not supposed to write VLD to 1 if it is already 1.

   The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

Figure 1483. MEMU error buffer uniqueness check flow chart

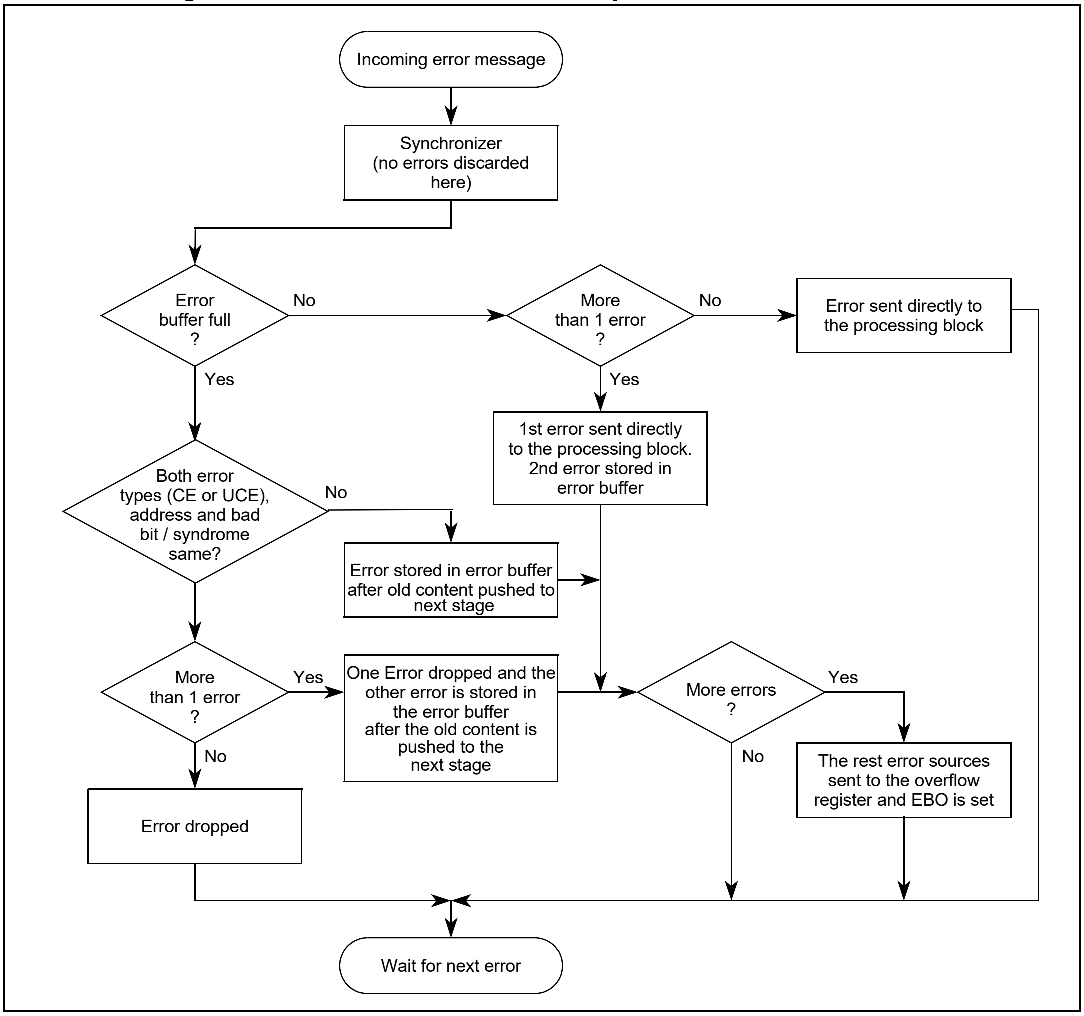
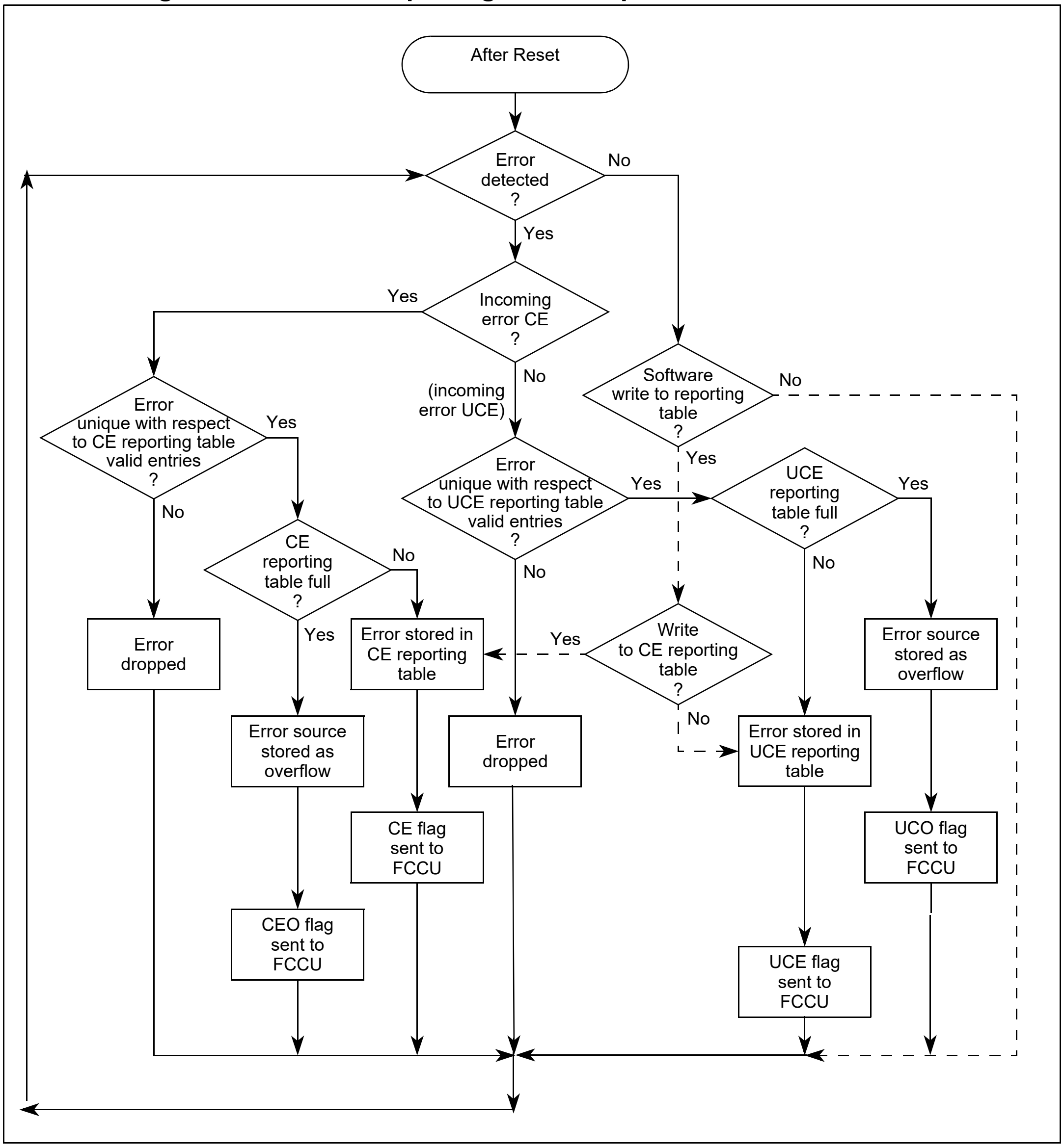


Figure 1484. MEMU reporting table uniqueness check flow chart



73.5 External signal description

MEMU has no external signals.

73.6 Memory map and register definition

73.6.1 Memory map

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

Note: The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

The Control, Error Flag and Debug registers are common for all MEMU error reporting blocks. The correctable error status, correctable error address, uncorrectable error status, uncorrectable error address and concurrent error overflow registers are replicated for each MEMU instance.

Note: The actual availability, number of registers and number of reported errors in the reporting table are chip-specific; see Chapter 7: Device configuration that describes how modules are configured and connected.

Table 1445. MEMU memory map

Offset or address	Register	Access	Reset value	Page
0x0000	Control register (CTRL)	R/W	0x0000_0000	Section 73.6.2.1
0x0004	Error Flag register (ERR_FLAG)	R/W	0x0000_0000	Section 73.6.2.2
0x000C	Debug register (DEBUG)	R/W	0x0000_0000	Section 73.6.2.3
0x0020 + *n**0x8 (*n* = 0 to 10 - 1)	System RAM correctable error reporting table status registers (SYS_RAM_CERR_STS*n*)	R/W	0x0000_0000	Section 73.6.2.4
0x0024 + *n**0x8 (*n* = 0 to 10 - 1)	System RAM correctable error reporting table address registers (SYS_RAM_CERR_ADDR*n*)	R/W	0x0000_0000	Section 73.6.2.5
0x0028 + (10 - 1)*0x8	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)	R/W	0x0000_0000	Section 73.6.2.6
0x002C + (10 - 1)*0x8	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)	R/W	0x0000_0000	Section 73.6.2.7
0x0030 + *n**0x4 + (10 - 1)*0x8 (*n* = 0 to 128÷32 - 1)(1)	System RAM concurrent overflow registers(2) (SYS_RAM_OFLW*n*)	w1c	0x0000_0000	Section 73.6.2.8
0x0620 + *n**0x8 (*n* = 0 to 2 - 1)	Peripheral RAM correctable error reporting table status registers (PERIPH_RAM_CERR_STS*n*)	R/W	0x0000_0000	Section 73.6.2.9
0x0624 + *n**0x8 (*n* = 0 to 2 - 1)	Peripheral RAM correctable error reporting table address registers (PERIPH_RAM_CERR_ADDR*n*)	R/W	0x0000_0000	Section 73.6.2.10
0x0628 + (2 - 1)*0x8	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)	R/W	0x0000_0000	Section 73.6.2.11
0x062C + (2 - 1)*0x8	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)	R/W	0x0000_0000	Section 73.6.2.12
0x0630 + *n**0x4 + (2 - 1)*0x8 (*n* = 0 to 32÷32 - 1)(1)	Peripheral RAM concurrent overflow registers(1) (PERIPH_RAM_OFLW*n*)	w1c	0x0000_0000	Section 73.6.2.13

Table 1445. MEMU memory map (continued)

Offset or address	Register	Access	Reset value	Page
0x0C20 + *n**0x8 (*n* = 0 to 20 - 1)	Flash memory correctable error reporting table status registers (FLASH_CERR_STS*n*)	R/W	0x0000_0000	*Section 73.6.2.14*
0x0C24 + *n**0x8 (*n* = 0 to 20 - 1)	Flash memory correctable error reporting table address registers (FLASH_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 73.6.2.15*
0x0C28 + (20 - 1)*8	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)	R/W	0x0000_0000	*Section 73.6.2.16*
0x0C2C + (20 - 1)*8	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)	R/W	0x0000_0000	*Section 73.6.2.17*
0x0C30 + *n**0x4 + (20 - 1)*0x8 (*n* = 0 to 5÷32 - 1)[1]	Flash memory concurrent overflow registers[1](FLASH_OFLW*n*)	w1c	0x0000_0000	*Section 73.6.2.18*

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.
2. The corresponding error sources for the Concurrent Overflow Register (OFLW) are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

73.6.2 Register descriptions

73.6.2.1 Control register (CTRL)

Offset: 0x0000 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		0	0	0	0	0	0	0	0	0	0	0	0	0	FLASH_ERR_SELECT	
W	SWR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1485. Control register (CTRL)

Table 1446. CTRL field descriptions

Field	Description
16 SWR	Software Reset bit. Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.
30:31 FLASH_ER R_SELECT	FLASH_ERR_SELECT[1:0] 00 Store 20xDEC (Default). Flash Double-bit error corrections only are stored as correctable errors. 01 Store 20xDEC or SEC. Flash Single-bit or Double-bit error corrections, both are stored as correctable errors. 10 Store 20xSEC. Flash Single-bit error corrections only are stored as correctable errors 11 Reserved

73.6.2.2 Error Flag register (ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO,F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a ‘1’. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Offset: 0x0004 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	PR_CE	PR_UCE	PR_CEO	PR_UCO	PR_EBO
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	F_CE	F_UCE	F_CEO	F_UCO	F_EBO	0	0	0	SR_CE	SR_UCE	SR_CEO	SR_UCO	SR_EBO
W				w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1486. Error flag register (ERR_FLAG)

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Table 1447. ERR_FLAG field descriptions

Field	Description
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag. Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to ‘1’ if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: – a new Flash ECC uncorrectable error was detected. – a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.

Table 1447. ERR_FLAG field descriptions (continued)

Field	Description
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to ‘1’ if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
27 SR_CE	System RAM ECC and MBIST Correctable Error detect flag. Indicates that a new and unique System RAM ECC and MBIST correctable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
28 SR_UCE	System RAM ECC and MBIST Uncorrectable Error Detect flag. Asserted when: – A new System RAM ECC and MBIST uncorrectable error is detected – A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
29 SR_CEO	System RAM ECC and MBIST Correctable error Overflow flag. Asserted when the System RAM ECC and MBIST correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.

Table 1447. ERR_FLAG field descriptions (continued)

Field	Description
30 SR_UCO	System RAM ECC and MBIST Uncorrectable error Overflow flag. Asserted when the System RAM ECC and MBIST uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECC and MBIST Error buffer Overflow. Flag asserted when the internal error processing buffer of System RAM ECC and MBIST MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to ‘1’ if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

73.6.2.3 Debug register (DEBUG)

This register is used to check the connections between MEMU & FCCU.

The error output of the MEMU towards FCCU shall stay set if forced until reset by SW via the MEMU in the normal way.

Offset 0x000C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	FR_PR_CE	FR_PR_UCE	FR_PR_CEO	FR_PR_UCO	FR_PR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	FR_F_CE	FR_F_UCE	FR_F_CEO	FR_F_UCO	FR_F_EBO	0	0	0	FR_SR_CE	FR_SR_UCE	FR_SR_CEO	FR_SR_UCO	FR_SR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1487. Debug register (DEBUG)

Table 1448. DEBUG field descriptions

Field	Description
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag going towards FCCU to 1.
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag going towards FCCU to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag going towards FCCU to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag going towards FCCU to 1.
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag going towards FCCU to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag going towards FCCU to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag going towards FCCU to 1.
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag going towards FCCU to 1.
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag going towards FCCU to 1.
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag going towards FCCU to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag going towards FCCU to 1.
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.

Table 1448. DEBUG field descriptions (continued)

Field	Description
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag going towards FCCU to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag going towards FCCU to 1.

73.6.2.4 System RAM correctable error reporting table status register (SYS_RAM_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0020 + *n**0x8 (*n* = 0 to (10 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0(1)	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write 1 to this bit may lead to non-zero invalid data in the fields 7:15

Figure 1488. System RAM correctable error reporting table status register (SYS_RAM_CERR_STS*n*)

Table 1449. SYS_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
24:31 BAD_BIT	Bad bit field Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

73.6.2.5 System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR*n*)

Offset: 0x0024 + *n**0x8 (*n* = 0 to (10 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1489. System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR*n*)

Table 1450. SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

73.6.2.6 System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0028 + (10 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	VLD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1490. System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Table 1451. SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

73.6.2.7 System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x002C + (10 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1491. System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Table 1452. SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

73.6.2.8 System RAM concurrent overflow registers (SYS_RAM_OFLW*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0030 + n*0x4 + (10 - 1)*0x8 (n = 0 to (128÷32 - 1))(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 1492. System RAM concurrent overflow registers (SYS_RAM_OFLW*n*)

Table 1453. SYS_RAM_OFLW*n* field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 73.7.3: Handling overflows (Multiple error reporting)* occurs.

73.6.2.9 Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STSn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0620 + n*0x8 (n = 0 to(2 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0(1)	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write 1 to this bit may lead to non-zero invalid data in the fields 7:15

Figure 1493. Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS*n*)

Table 1454. PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
24:31 BAD_BIT	Bad bit field Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

73.6.2.10 Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR*n*)

Offset: 0x0624 + *n**0x8 (*n* = 0 to (2 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1494. Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR*n*)

Table 1455. PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

73.6.2.11 Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0628 + (2 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1495. Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Table 1456. PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

73.6.2.12 Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x062C + (2 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1496. Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Table 1457. PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

73.6.2.13 Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLW*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0630 + *n**0x4 + (2 - 1)*0x8 (*n* = 0 to (32÷32 - 1))(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 1497. Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLW*n*)

Table 1458. PERIPH_RAM_OFLW*n* field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 73.7.3: Handling overflows (Multiple error reporting)* occurs.

73.6.2.14 Flash memory correctable error reporting table status register (FLASH_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

MEMU captures flash ecc error address on 128bit boundary.

Offset: 0x0C20 + *n**0x8 (*n* = 0 to (20 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0(1)	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	BAD_BIT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write 1 to this bit may lead to non-zero invalid data in the fields 7:15

Figure 1498. Flash memory correctable error reporting table status register (FLASH_CERR_STS*n*)

Table 1459. FLASH_CERR_STS*n* register field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
16:31 BAD_BIT	Bad bit field Indicates the bit position in FLASH where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFFFF. Any value other than 0xFFFF - Position of the bit in FLASH where error is found. 0xFFFFInvalid Bad Bit. This field should not be used when processing errors.

73.6.2.15 Flash memory correctable error reporting table address register (FLASH_CERR_ADDR*n*)

Offset: 0x0C24 + *n**0x8 (*n* = 0 to (20 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1499. Flash memory correctable error reporting table address register (FLASH_CERR_ADDR*n*)

Table 1460. FLASH_CERR_ADDR*n* register field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

73.6.2.16 Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C28 + (20 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1500. Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

Table 1461. UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

73.6.2.17 Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C2C + (20 - 1)*8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1501. Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Table 1462. FLASH_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

73.6.2.18 Flash memory concurrent overflow registers (FLASH_OFLWn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C30 + *n**0x4 + (20 - 1)*0x8 (*n* = 0 to (5÷32 - 1))(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 1502. Flash memory concurrent overflow registers (FLASH_OFLWn)

Table 1463. FLASH_OFLWn field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 73.7.3: Handling overflows (Multiple error reporting)* occurs.

73.7 Functional description

In this section, the term “bad bit” and “syndrome” are equivalent, and are used to indicate the position of the bit in error.

73.7.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known errors which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit = ‘1’ for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address.The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows:

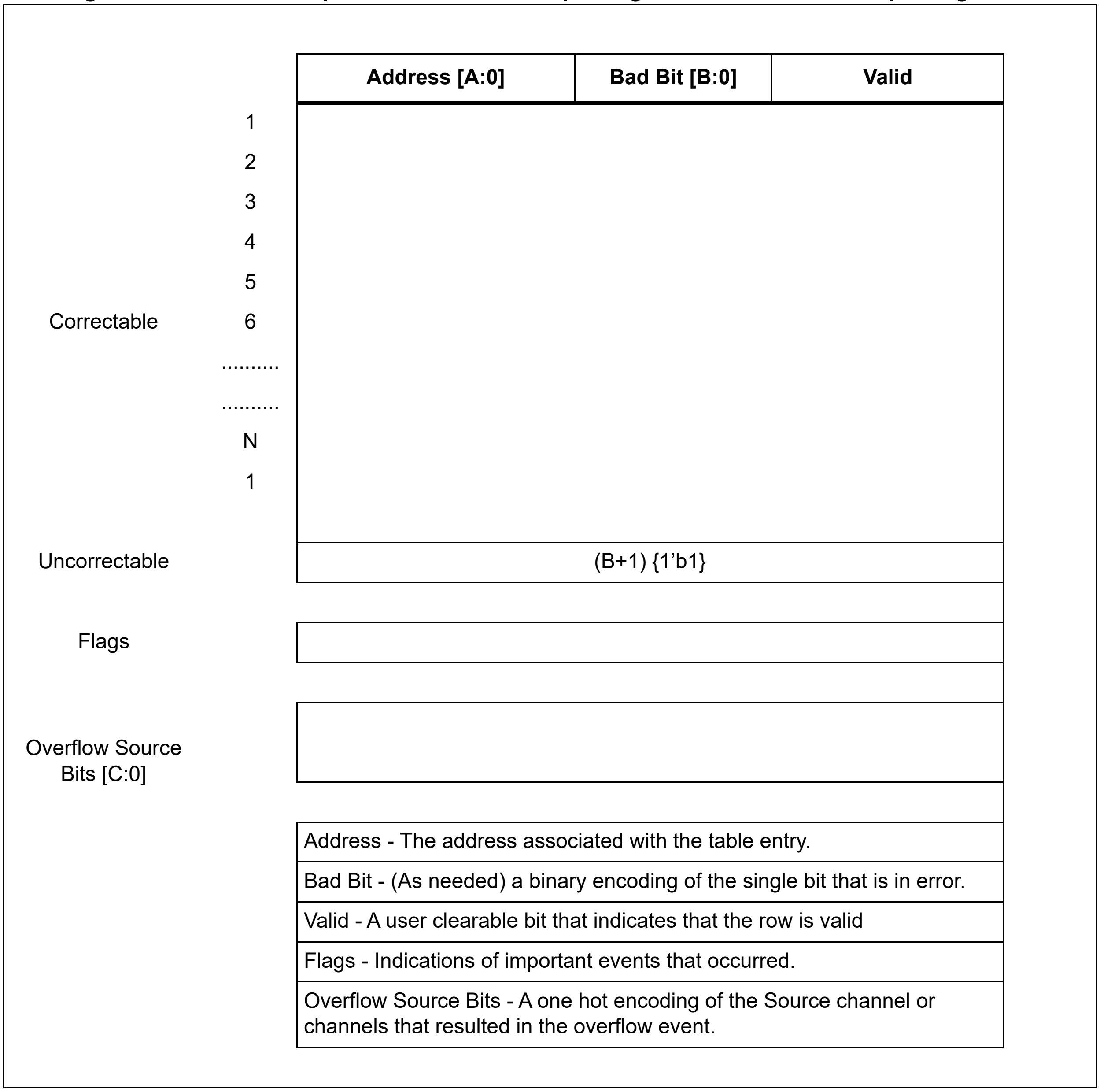
1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any error handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU writes an error with the same address that Software wants to program, both the errors will be stored in the reporting table (duplicate entries).

73.7.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To in-validate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.

Figure 1503. Generic representation of the reporting table of one MEMU reporting block



73.7.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

1. Error buffer overflow–More than two memories reporting an error at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
   – More than 2 errors detected at the same time. This would lead to overflow in the error buffer. MEMU can process only one error at a time while storing the other in the error buffer. So this would be indicated as an overflow. The uniqueness check in the input buffer does not “guarantee” that an ECC-supervised memory with only one error will not be marked as overflow source. If that one error occurs together with two (or more) additional errors at the same cycle it can get flagged as an overflow.
2. Correctable/Uncorrectable Error Overflow–Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
   – An ECC error which matches the address of an MBIST error is dropped (same as if it matched the address of an ECC error). In case, where a MBIST error comes with ECC error already in the table, both the address and bad bits are compared to determine uniqueness.
3. The Overflows will be stored in the bit-wise order which means that if the error bit 31 reports an error, the bit 31 of the overflow register will be set to ‘1’.

74 Indirect Memory Access (IMA)

74.1 Introduction

Indirect Memory Access (IMA) refers to the activity of accessing any chip memory for the purpose of reading or modifying data, or both, including ECC check bits. This capability is useful for test activities, for example, verifying ECC functionality.

The SPC584Bx MCU provides two mechanisms for IMA:

- The built-in CPU End2End ECC test capability provides a path for accessing data and ECC check bits from some SRAM arrays.
- An IMA controller module provides registers for selecting, reading, and writing memory data.

Note: Not all SRAM data and ECC bits are accessible. Refer to the Device Configuration chapter for details.

74.2 Accessing memory via CPU End2End ECC test

The processor cores can access all bits (including ECC check bits) in SRAM arrays that support E2E ECC with the exceptions of addresses that are blocked by MPU configuration.

Note: All memory accesses by the processor cores are filtered through the MPU. There are no backdoors to blocked memory addresses.

SRAM is selected by absolute memory address with check bit data in a special register. The register used depends on which memory is being accessed: I-cache/D-cache, or other memory.

Since all CPU memory accesses are arbitrated by the crossbar (XBAR), there is no need to halt normal memory access in software.

Memory is accessed via processor core registers. The register(s) used depends on the memory being accessed. This is described in the following sections.

74.2.1 Accessing I-cache and D-cache memory via processor core

For I-cache and D-cache memory, the SRAM location is selected by cache way, set and word with data and check bits in registers provided specifically for reading and writing cache tag bits and ECC check bits. There are two Device Control Registers (DCRs) provided for cache access:

- Cache Debug Access Control register (CDACNTL)
- Cache Debug Access Data register (CDADATA)

These registers are summarized in the following sections.

74.2.1.1 Cache Debug Access Control register (CDACNTL)

The Cache Debug Access Control register (CDACNTL) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to I-cache and D-cache memory.

0	1:2	3	4:5	6:12	13:15	16:23	24:27	28	29	30:31
T/D	0	CWAY	0	CSET	WORD	PARITY CHECK BITS	0	CACHE	R/W	GO
DCR - 351; Read/Write; Reset - 0x0										

Figure 1504. Cache Debug Access Control register (CDACNTL)

The CDACNTL bits are described in *Table 1464*.

Table 1464. CDACNTL field descriptions

Bits	Name	Description
0	T/D	Tag / Data: 0 Data array selected 1 Tag array selected
1:2	—	Reserved - read as zeros.
3	CWAY	Cache Way: Specifies the cache way to be selected.
4:5	—	Reserved - read as zeros.
6:12	CSET	Cache Set: Specifies the cache set to be selected. Only valid set numbers for the selected cache are supported, otherwise results are undefined.
13:15	WORD	Word (Used for data array access only, I or D-cache) Specifies one of eight words of selected set.
16:23	PARITY CHECK BITS	Parity check bits (I or D-cache): D-cache Data array, I-cache Data Array: parity check bits for data. Tag Array: parity check bits for tag.
24:27	—	Reserved - read as zeros.
28	CACHE	Cache Select: Specifies the cache to be selected. 0 Selects the data cache for the operation. 1 Selects the instruction cache for the operation.
29	R/W	Read / Write: 0 Selects write operation. Write the data in the CDADATA register to the location specified by this CDACNTL register. 1 Selects read operation. Read the cache memory location specified by this CDACNTL register and store the resulting data in the CDADATA register and store the parity bits in this CDACNTL register.
30:31	GO	GO command bits: 00 Inactive or complete (no action taken) hardware sets GO=00 when an operation is complete 01 Read or write cache memory location specified by this CDACNTL register. 1x Reserved

74.2.1.2 Cache Debug Access Data register (CDADATA)

The Cache Debug Access Data register (CDADATA) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to I-cache and D-cache memory.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TAG or DATA																															
DCR - 350; Read/Write; Reset - Undefined/Unaffected																															

Figure 1505. Cache Debug Access Data register (CDADATA)

The CDADATA bits are described in *Table 1465*.

Table 1465. CDADATA field descriptions

Bits	Name	Description
0:31	TAG	TAG Array Access Data - when accessing the tag array of the I-cache: 0–19 Tag compare bits 20–22 Reserved 23 Valid bit 24 R bit 25 SO bit 26–28 Reserved 29–31 Lockout bits. These three bits should have the same value, 1- Locked out, 0- not locked out. TAG Array Access Data - when accessing the tag array of the D-cache: 0–20 Tag compare bits 21–22 Reserved 23 Valid bit 24 R bit 25 SO bit 26 SW bit 27 UW bit 28 Reserved 29–31 Lockout bits. These three bits should have the same value, 1-Locked out, 0- not locked out.
	DATA	DATA Array Access Data (Bytes 0:3 of the selected word) - when accessing the data array of either cache: 0–7 byte 0 8–15 byte 1 16–23 byte 2 24–31 byte 3

74.2.2 Accessing non I-cache or D-cache memory via processor core

The e2eECC Error Control/Status register 0 (E2EECSR0) is provided specifically for reading and writing E2E ECC check bits. This register is described in the following section.

74.2.2.1 e2eECC Error Control/Status register 0 (E2EECSR0)

The e2eECC Error Control/Status register 0 (E2EECSR0) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to E2E ECC data. It is shown in the following figure.

0–3	4–11	12–15	16–17	18–19	20–23	24–31
0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
DCR - 511; Read/Write; Reset - 0x0; Privileged						

Figure 1506. e2eECC Error Control/Status register 0 (E2EECSR0)

The E2EECSR0 bits are described in *Table 1466*.

Table 1466. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros.
4:11	RCHKBIT	Read Checkbits: This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the data BIU. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation, assuming interrupts are masked.
12:15	—	Reserved - read as zeros.
16:17	WRC	Write Control: 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) with the values in the WCHKBIT control field. Software must clear this field before re-writing to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10, by driving the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved **Note:** Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control: 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying p_hwchkbit[7:0] based on CHKINVT. Software must clear this field before re-writing to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying p_hwchkbit[7:0] based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved **Note:** Checkbit substitution has priority over Error injection

Table 1466. E2EECSR0 field descriptions (continued)

Bits	Name	Description
20:23	—	Reserved - read as zeros.
24:31	WCHKBIT/CHKINVT	Write Checkbits / Checkbit Invert Mask: This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out when error injection occurs.

74.3 Accessing memory via the IMA module

The Indirect Memory Access (IMA) module provides an alternative way to access on-chip memory without going through standard peripheral or system RAM interfaces to check the data or change the data.

This alternative path can be used for diagnostic purposes. For example, for SRAM arrays that do not support E2E ECC, the IMA block provides a way to write corrupted ECC values so that the ECC error decoding logic can be checked.

Note the following restrictions on memory access via IMA:

- When a bus master performs an IMA access, no other bus master shall perform an access to the same memory.
- Before any IMA access, the user shall disable the FCCU fault related to the IMA. If not, the FCCU triggers a fault.
- The IMA must be unlocked for reads.

The order of precedence for access to a memory is shown below. Once the IMA has been granted access, no other accesses are allowed until the IMA select line to the selected memory is negated by the IMA.

Note: Once the IMA has selected a RAM, it expects no other accesses from any other master until deselected by the IMA.

The IMA module signals the FCCU when it is reading or writing to a memory address. This enables the FCCU to assess whether the access is intentional or spurious. For an intentional access the respective FCCU error input needs to be disabled, otherwise it should be enabled.

74.3.1 Features

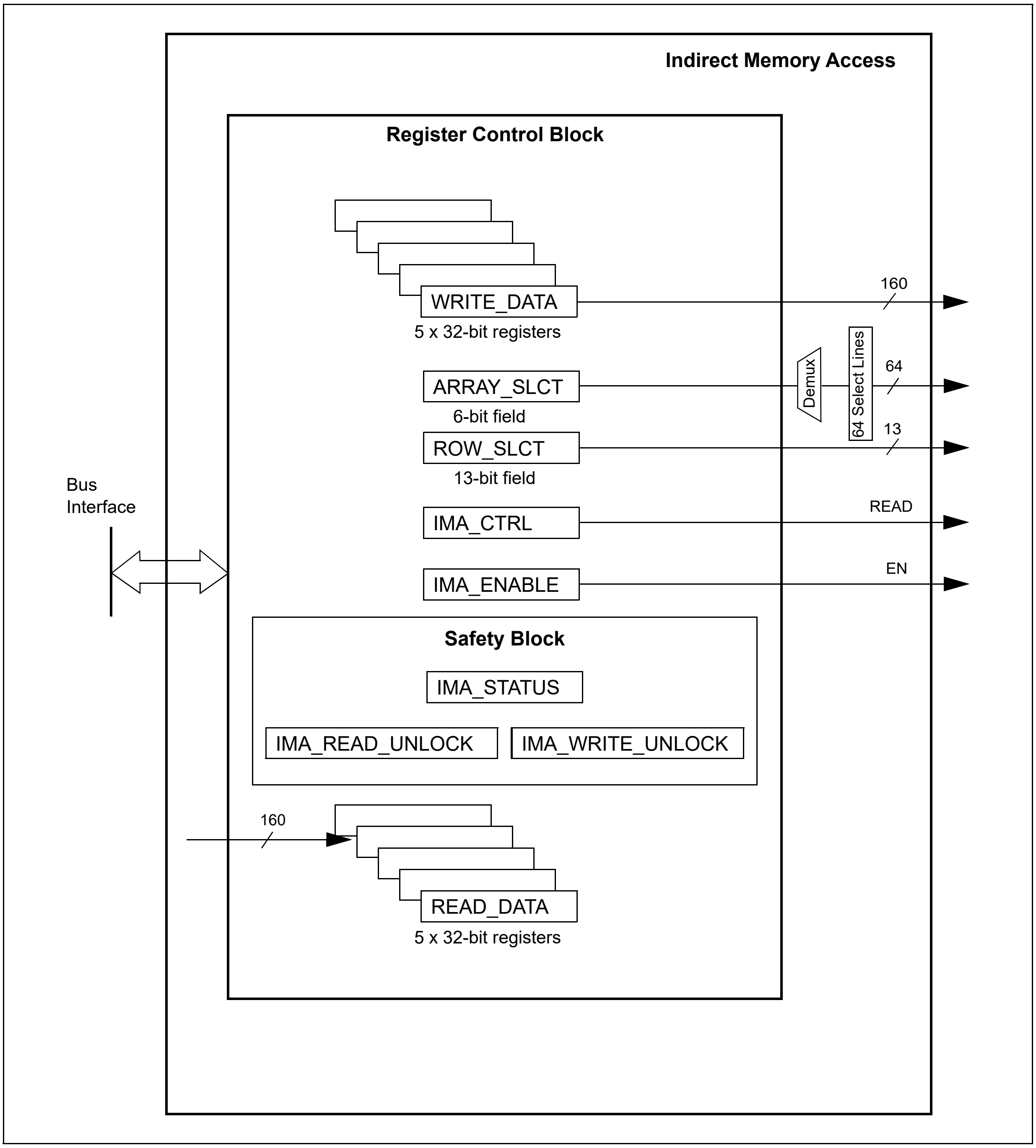
The IMA module includes the following features:

- Able to read all bits in all RAM arrays connected to the IMA, once the IMA is unlocked for reads.
- Enables reading and manipulation of ECC check bits.
- During an IMA SRAM access, other accesses are affected as follows:
  - Reads return random data
  - Writes have no effect
- Any IMA access to memory are indicated in the Fault Collection and Control Unit (FCCU).
- IMA accesses to memory do not bypass any SRAM repair which may have occurred during production test of the device.
- IMA accesses to memory use a physical address, not a logical address, and thus memory addresses used by the IMA are not memory mapped.

74.3.2 Block diagram

The following figure shows the structure and function of the IMA module. The Register Control Block contains the memory-mapped registers accessible by software. The Safety Block prevents spurious activation of the IMA module. It ensures the IMA does not access RAM unless the system needs it too.

Figure 1507. IMA block diagram



74.3.3 IMA Module Memory Map and Registers

This section provides details of the IMA memory map. It also provides detailed descriptions for each of the registers in that map.

The Indirect Memory Access unit's memory map allows for the CPU to access RAM's on the chip to write test data so that single-bit ECCs and multi-bit ECC logic can be tested.

Table 1467. IMA memory map

Offset Address	Register	Section
0x0000	IMA Control register (IMA_CTRL)	Section 74.3.3.1
0x0004	IMA Enable Access register (IMA_ENABLE)	Section 74.3.3.2
0x0008	IMA Status register (IMA_STATUS)	Section 74.3.3.3
0x000C	IMA RAM SELECT register (IMA_SLCT)	Section 74.3.3.4
0x0010	IMA Write Unlock register (IMA_WRITE_UNLOCK)	Section 74.3.3.5
0x0014	IMA Read Unlock register (IMA_READ_UNLOCK)	Section 74.3.3.6
0x0018–0x002B	Reserved	
0x002C	IMA RAM Write Data register 4 (IMA_WRITE_DATA_4)	Section 74.3.3.7
0x0030	IMA RAM Write Data register 3 (IMA_WRITE_DATA_3)	Section 74.3.3.7
0x0034	IMA RAM Write Data register 2 (IMA_WRITE_DATA_2)	Section 74.3.3.7
0x0038	IMA RAM Write Data register 1 (IMA_WRITE_DATA_1)	Section 74.3.3.7
0x003C	IMA RAM Write Data register 0 (IMA_WRITE_DATA_0)	Section 74.3.3.7
0x0040–0x004B	Reserved	
0x004C	IMA RAM Read Data register 4 (IMA_READ_DATA_4)	Section 74.3.3.8
0x0050	IMA RAM Read Data register 3 (IMA_READ_DATA_3)	Section 74.3.3.8
0x0054	IMA RAM Read Data register 2 (IMA_READ_DATA_2)	Section 74.3.3.8
0x0058	IMA RAM Read Data register 1 (IMA_READ_DATA_1)	Section 74.3.3.8
0x005C	IMA RAM Read Data register 0 (IMA_READ_DATA_0)	Section 74.3.3.8

74.3.3.1 IMA Control register (IMA_CTRL)

The IMA CTRL register controls read/write accesses. Since the IMA may interfere with proper read/write access of RAMs, a locking scheme is implemented to prevent accidental activation of the IMA. Using this scheme, the IMA read/write accesses can be locked when desired.

Offset: 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1508. IMA Control register (IMA_CTRL)

Table 1468. IMA_CTRL field descriptions

Field	Description
31 READ	IMA Read This bit controls whether the IMA will perform a read or write 0 The IMA access will be a write. 1 The IMA access will be a read.

74.3.3.2 IMA Enable Access register (IMA_ENABLE)

The Enable Access Register is a single-bit register that starts an IMA access. The bit automatically clears itself after two clocks thus providing an enable pulse.

Note: Back to back writes to the enable bit is not valid. It will still generate a two-clock enable pulse and will be considered as a single access initiation.

Offset: 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1509. IMA Enable Access register (IMA_ENABLE)

Table 1469. IMA_ENABLE field descriptions

Field	Description
31 EN	IMA Enable This bit controls RAM access initiation by the IMA. Once set, this bit is cleared after two clocks to provide an enable pulse. 0 No IMA access will occur or the IMA has finished the previous access. 1 Start an IMA access or an IMA access is occurring.

74.3.3.3 IMA Status register (IMA_STATUS)

The IMA Status register shows the current status of the IMA as to which operations are unlocked.

Offset: 0x08 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	WRITE_LOCK	0	0	0	0	0	0	0	READ_LOCK
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 1510. IMA Status register (IMA_STATUS)

Table 1470. IMA_STATUS field descriptions

Field	Description
23 WRITE_LOCK	The ability of the IMA to do write accesses. The WRITE_LOCK bit shows the current status of the lock on IMA write accesses. If set, the IMA does not write to the memories. If cleared, the IMA is allowed to do writes to the memories. 0 Write accesses from the IMA to memories are allowed. 1 Write accesses from the IMA to memories are not allowed. Any memory writes from the IMA are ignored.
31 READ_LOCK	The ability of the IMA to do read accesses. The READ_LOCK bit shows the current status of the lock on IMA read accesses. If set, the IMA does not do reads from the memories. If cleared, the IMA is allowed to do reads from the memories. 0 Read accesses from the IMA to memories are allowed. 1 Read accesses from the IMA to memories are not allowed. Any memory reads from the IMA are ignored.

74.3.3.4 IMA RAM Select register (IMA_SLCT)

The IMA RAM Select register provides a method of selecting which RAM memory block will be accessed. This method is used because RAM is to be accessed in a non-functional way, such as accessing all bits including the ECC, and accessing on RAM word boundaries instead of byte or 32-bit word boundary.

Note: Only one RAM can be selected at a time. Each SoC has a different encoding for the RAMs. Refer to the Device Configuration chapter for details.

Offset: 0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	ROW_SLCT												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ARRAY_SLCT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1511. IMA RAM SELECT register (IMA_SLCT)

Table 1471. IMA_SLCT field descriptions

Field	Description
3:15 ROW_SLCT	The ROW_SLCT value specifies the row of the current RAM array under test to be selected. Hence the largest RAM which the IMA can access is 32 KB RAM.
26:31 ARRAY_SLCT	The ARRAY_SLCT specifies the RAM array to be tested. Up to 64 RAM instances can be supported.

74.3.3.5 IMA Write Unlock register (IMA_WRITE_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The WRITE_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for writes.

Offset: 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WRITE_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRITE_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1512. IMA Write Unlock register (IMA_WRITE_UNLOCK)

Table 1472. IMA_WRITE_UNLOCK field descriptions

Field	Description
0:31 WRITE_KEY	Key to unlock the IMA. – 0x04A43F95 is the first key that has to be written to start the unlock process. – 0xE4A9EBF7 is the second key that has to be written to unlock the IMA.

74.3.3.6 IMA Read Unlock register (IMA_READ_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The READ_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for reads.

Offset: 0x14 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	READ_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	READ_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1513. IMA Read Unlock register (IMA_READ_UNLOCK)

Table 1473. IMA_READ_UNLOCK field descriptions

Field	Description
0:31 READ_KEY	Key to unlock the IMA Read Access to RAMs. – 0xF06AB5BC is the first key that has to be written to start the unlock process. – 0x14081B56 is the second key that has to be written to unlock the IMA.

74.3.3.7 IMA RAM Write Data register n (IMA_WRITE_DATA_*n*)

The IMA_WRITE_DATA_*n* registers hold the data values to be written to the RAM. The number of active data bits depends on the number of bits for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a write access.

Offset: 0x2C (WRITE_DATA_4)
0x30 (WRITE_DATA_3)
0x34 (WRITE_DATA_2)
0x38 (WRITE_DATA_1)
0x3C (WRITE_DATA_0)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WRITE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRITE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1514. IMA RAM Write Data register *n* (IMA_WRITE_DATA_*n*)

Table 1474. IMA_WRITE_DATA_*n* field descriptions

Field	Description
0:31 WRITE	Contains data to be written to RAM. – WRITE_DATA_4[WRITE] contains write data bits[159:128] – WRITE_DATA_3[WRITE] contains write data bits[127:96] – WRITE_DATA_2[WRITE] contains write data bits[95:64] – WRITE_DATA_1[WRITE] contains write data bits[63:32] – WRITE_DATA_0[WRITE] contains write data bits[31:0]

74.3.3.8 IMA RAM Read Data register n (IMA_READ_DATA_*n*)

The IMA_READ_DATA_*n* registers hold the data values read from the RAM. The number of active data bits depends on the number of bits for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a read access. Performing a read operation on IMA_READ_DATA_*n* registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Offset: 0x4C (READ_DATA_4)
0x50 (READ_DATA_3)
0x54 (READ_DATA_2)
0x58 (READ_DATA_1)
0x5C (READ_DATA_0)

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	READ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	READ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1515. IMA RAM Read Data register n (IMA_READ_DATA_*n*)

Table 1475. IMA_READ_DATA_*n* field descriptions

Field	Description
0:31 READ	Contains data to be written to RAM. – READ_DATA_4[READ] contains read data bits[159:128] – READ_DATA_3[READ] contains read data bits[127:96] – READ_DATA_2[READ] contains read data bits[95:64] – READ_DATA_1[READ] contains read data bits[63:32] – READ_DATA_0[READ] contains read data bits[31:0]

74.3.4 IMA module functional description

This section describes the operation of the IMA, beginning with the unlocking of reads and writes, the software initialization, then the software interface for generating single-bit ECC errors, generating multiple-bit ECC errors, and a software supported memory BIST.

74.3.4.1 Initialization sequence

This section describes which registers are reset due to hardware reset and what locations the user must initialize prior to doing an access by the IMA.

74.3.4.2 Hardware controlled initialization

In the IMA, registers and control logic are reset by hardware (system reset). A system reset deasserts output signals and resets general configuration bits.

74.3.4.3 User initialization (prior to asserting IMA_ENABLE[EN])

The user needs to initialize portions of the SoC prior to setting the IMA_ENABLE[EN] bit. The exact values depend on the particular application. Refer to the specific chapter's memory map to see how to turn off that peripheral so that accesses are not happening at the same time as the IMA. Without the peripheral shut down, there is a small chance that the IMA and the peripheral can interfere with each other in accessing the RAM.

74.3.4.3.1 Shut down the peripheral

The IMA can cause problems with the system or the peripheral if both the IMA and the other system components are accessing the RAM during normal operation.

To shut down the peripheral, read the peripheral's documentation. The documentation will describe how to put the peripheral into a quiet state so that it will not access the RAM during the IMA access. If the other functional path is running at the same time, the data for the other peripheral will be corrupted and unknown.

74.3.4.3.2 Unlocking reads

The IMA will not allow a read to any memory unless it has been unlocked. This is a safety feature so that it will take two unique writes to the READ_KEY register to unlock writes. To unlock the read feature, the system will have to do the following:

1. Write 0xF06AB5BC to READ_KEY.
2. Write 0x14081B56 to READ_KEY.

The IMA has two features to make the READ_KEY lock in the following instance:

- Writing an incorrect key to READ_KEY.

74.3.4.3.3 Unlocking writes

The IMA will not allow a write to any memory unless it has been unlocked. This is a safety feature so that it will take two unique writes to the WRITE_KEY register to unlock writes. To unlock the write feature, the system will have to do the following:

1. Write 0x04A43F95 to WRITE_KEY.
2. Write 0xE4A9EBF7 to WRITE_KEY.

The IMA has a feature to make the WRITE_KEY lock in the following two instances:

1. Writing an incorrect key to WRITE_KEY.
2. Not setting the IMA_ENABLE[bit] for 16,384 system clocks.

74.3.4.4 IMA lock

The IMA module also allows for read/write access locking via an input pin. If this input is asserted, both read/write accesses via IMA are locked. This locking scheme overrides the above KEY based locking scheme. Hence it provides additional safety over the KEY based locking method. However, usage of this feature is entirely SOC specific.

74.3.4.5 IMA access

There are two differences between a read access and a write access. The first difference is that the IMA_STATUS[WRITE_LOCK] bit has to be cleared for writes and the IMA_STATUS[READ_LOCK] has to be cleared for reads. The second difference is that the IMA_CTRL[READ] has to be set for reads and IMA_CTRL[READ] has to be cleared for writes.

The following sections show the different accesses that are available depending on the options.

74.3.4.5.1 Read access

Read Accesses are expected to follow the sequence below:

1. Unlock Read Access by performing appropriate consecutive writes to READ_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Set the READ bit in the IMA_CTRL register.
4. Set the EN bit in the IMA_ENABLE register to initiate read access. The EN bit will self-clear after two clocks.
5. Software must wait an appropriate number of clocks depending on the speed of the SRAM for the access to complete.
6. Read data from READ_DATA registers. This causes data from RAM to be latched into the IMA and returned to the CPU on the next clock.

74.3.4.5.2 Write access

Write Accesses are expected to follow the sequence below:

1. Unlock Write Access by performing appropriate consecutive writes to WRITE_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Write the Write Data in WRITE_DATA registers.
4. Clear the READ bit in the IMA_CTRL register
5. Set the EN bit in the IMA_ENABLE register to initiate write access. The EN bit will self-clear after two clocks.
6. Software must wait an appropriate number of clocks for the IMA write to occur, depending on the speed of the SRAM being written.

74.3.5 Application information

Caution: The IMA will inhibit correct operation of the underlying peripherals. The application software is responsible for ensuring there is not a resource conflict.

The primary function of the IMA module is to provide a means for the system to program invalid ECC bits and see that the appropriate peripheral is able to detect the error. For a single-bit error, the peripheral should be able to correct the error. For a multiple-bit error, the peripheral should report an error.

75 Fault Collection and Control Unit (FCCU)

75.1 Introduction

The Fault Collection and Control Unit (FCCU) offers a systematic approach to fault detection and control. The FCCU provides a hardware channel to collect errors and to place the device into a safety state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

75.2 Features

The FCCU offers a systematic approach to fault detection and control. The distinctive features of the module are:

- 1 to 128 recoverable faults management
- HW or SW fault recovery management
- Fault detection collection
- Fault injection (fake faults; refer to the implementation-specific details)
- Bi-stable, Dual Rail and Time Switching output protocols on EOUT
- Watchdog timer for the reconfiguration phase
- Redundant collection of hardware checker (for example RCCU) results
- Redundant collection of error information from safety relevant modules on the device
- Collection of test results
- Configurable and graded fault control
- Internal (SoC) reactions
  - ALARM state: interrupt request
  - FAULT state: long functional reset request pulse, short functional reset request pulse, NMI
- Internal reactions (independently configurable for each RF)
  - No reset reaction
  - IRQ
  - Short functional reset
  - Long functional reset
  - NMI
- External reaction (failure is reported to the outside world via one or more output pins)
- External reaction (fault state): EOUT signaling. Failure indication via the pin(s) is controlled by the FCCU
- Configuration lock: The FCCU's configuration is lockable in two ways
  - Permanently locked until next reset or transiently locked until a specific key is written. FCCU gets transiently locked again if an invalid key is written into

FCCU_TRANS_LOCK register (a value different from 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register

- One of the failure indication pins is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the failure indication protocol is configured to be a toggling protocol that is dual rail and time switching
- After power-on the error out pins have high impedance[u]. FCCU goes to operational state only on software request
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in bi-stable mode only). T_min = 250 µs + delta_T, with delta_T parameter being configurable by SW up to 10 ms
- The overall error detection, processing and indication time is less than 10 ms
- The actual maximum time is five safe (IRCOSC) clock cycles

Two classes of fault are identified based on the criticality and the related reactions. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable based on the fault criticality. The default configuration can be modified only in a specific FCCU state for application/test/debugging purposes. FCCU is designed for assuming that the system clock is faster than the IRC clock.

75.2.1 Standard features

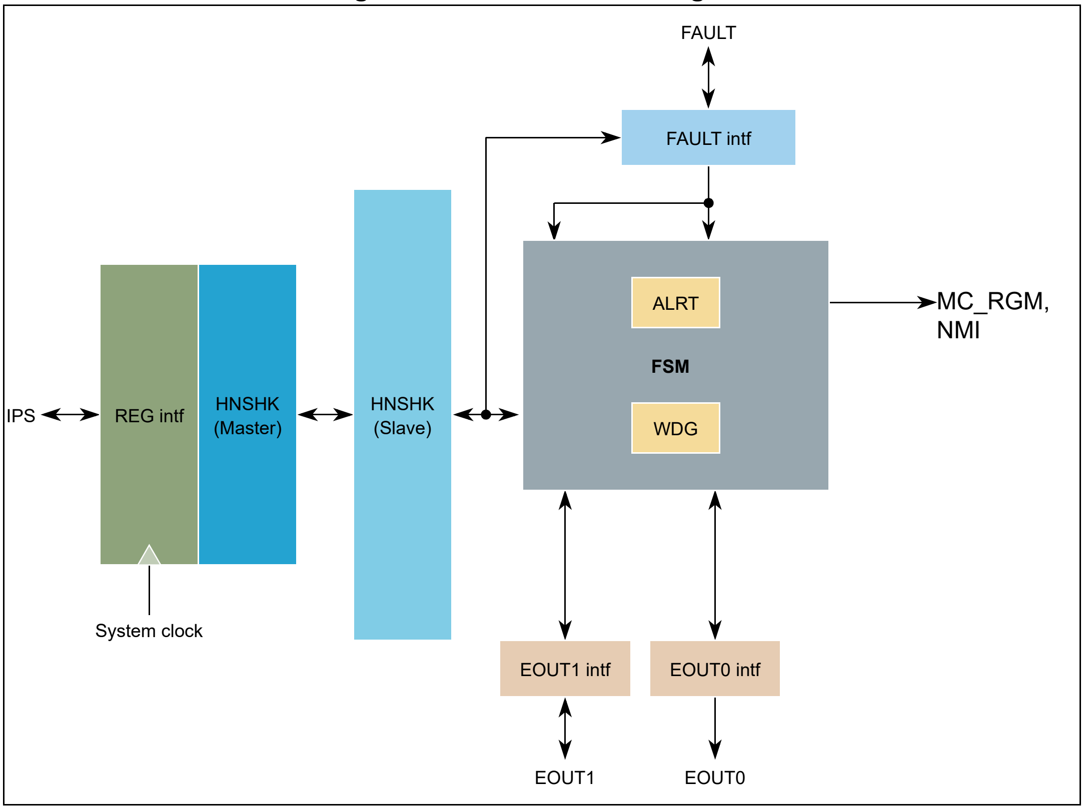
- IPS (slave bus signals) bus interface

75.3 Block diagram

The top-level diagram of the FCCU module is given in *Figure 1516*.

u. Actual value depends on the SoC settings at pad level.

Figure 1516. FCCU block diagram



The FCCU module includes the submodules listed in *Table 1476*.

Table 1476. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface
HNSHK blocks (master and slave blocks)	Includes the FSMs to support the handshake between the REG if and the FSM unit due to the usage of two asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: Watchdog timer (WDG) Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

75.4 Signal description

75.4.1 Pinout

The FCCU generates two external signals, EOUT[0] and EOUT[1]. These are described in *Section 75.5.9: EOUT interface*.

For the availability of these signals on a chip, refer to the device configuration chapter.

75.4.2 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported:

- word (32 bits) data write/read operations to any registers.
- low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers.
- byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0] data write/read operations to any registers.
- any other operation (free byte enables or other operations) must be avoided.

The FCCU module generates a transfer error in the following cases:

- any write/read access executed outside the register address space of the peripheral. Note that if some registers are not implemented (for example FCCU_CFS_CFGx registers), they generate no bus transfer error when accessed in writing.
- any write/read operation different from byte/hword/word (free byte enables or other operations) on each register.
- any write access executed on configuration registers, when the fccu is not in the CONFIG state. Configuration registers are those registers which can be written only in CONFIG state. Refer to *Table 1485: FCCU memory map*.
- any write executed on FCCU_CFG_TO register, when the fccu is in the CONFIG state.
- any write access to the Transient and permanent Lock Register in any mode other than supervisor access mode.

The registers of the FCCU module are accessible (read/write) in each access mode: user, supervisor or test.

75.5 Functional description

75.5.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is asserted. That is, if 0 on the fault signal indicates fault, then the status flags are valid until the fault line is ‘0’. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched in an external module at the FCCU. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the error is recoverable by itself when the error condition gets corrected.

The following types of reset are applicable:

- Destructive reset: any type of reset related to a power failure condition that implies a complete system re-initialization.
- Long functional reset: implies the Flash and digital circuitry (most of it except FCCU, STCU) initialization.
- Short functional reset: implies the digital circuitry (most of it except FCCU, STCU) initialization.

75.5.2 FSM description

The FCCU module functionality is depicted by the FSM diagram given in *Figure 1517*.

Basically four states are identified with the following meaning:

- CONFIG: The configuration state is used only to modify the default configuration of the FCCU. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, recoverable fault masking) can be accessed in write mode only in the CONFIG state.
  The CONFIG state is accessible only in NORMAL state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset of the FCCU, the transient lock register is unlocked by writing 0xBC into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (a value different from 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register. After reset lifts, the state shall be transiently locked (permanent lock → is unlocked and transient lock → is locked).
  The CONFIG to NORMAL state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to NORMAL appears at the same time, watchdog timeout had the priority and hence the Configuration registers (those that are writable only in CONFIG mode) are reset to their default values. Also, FCCU_CFG_TO and FCCU_EINOUT are reset to their default values, although they do not meet the condition of being a configuration register. The movement to NORMAL is made.
  The incoming faults, occurring during the configuration phase (CONFIG state) are anyway latched in order to process them when the FCCU is moved into the NORMAL state, according to the new configuration.
- NORMAL: This is the FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Succeeding one of the following events:
  - unmasked recoverable faults with the timeout disabled → the FCCU moves to the FAULT state.
  - unmasked recoverable faults with the timeout enabled → the FCCU moves to the ALARM state.
  - masked recoverable faults → the FCCU stays in NORMAL state.
- ALARM: the FCCU moves into the ALARM state when an unmasked recoverable fault occurs and the timeout is enabled. The transition to the ALARM state goes along with an interrupt request (ALARM state), if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to the

FAULT state. The timeout is re-initialized if the FCCU state moves to the NORMAL state. The timeout restarts following the recovery from the FAULT state.

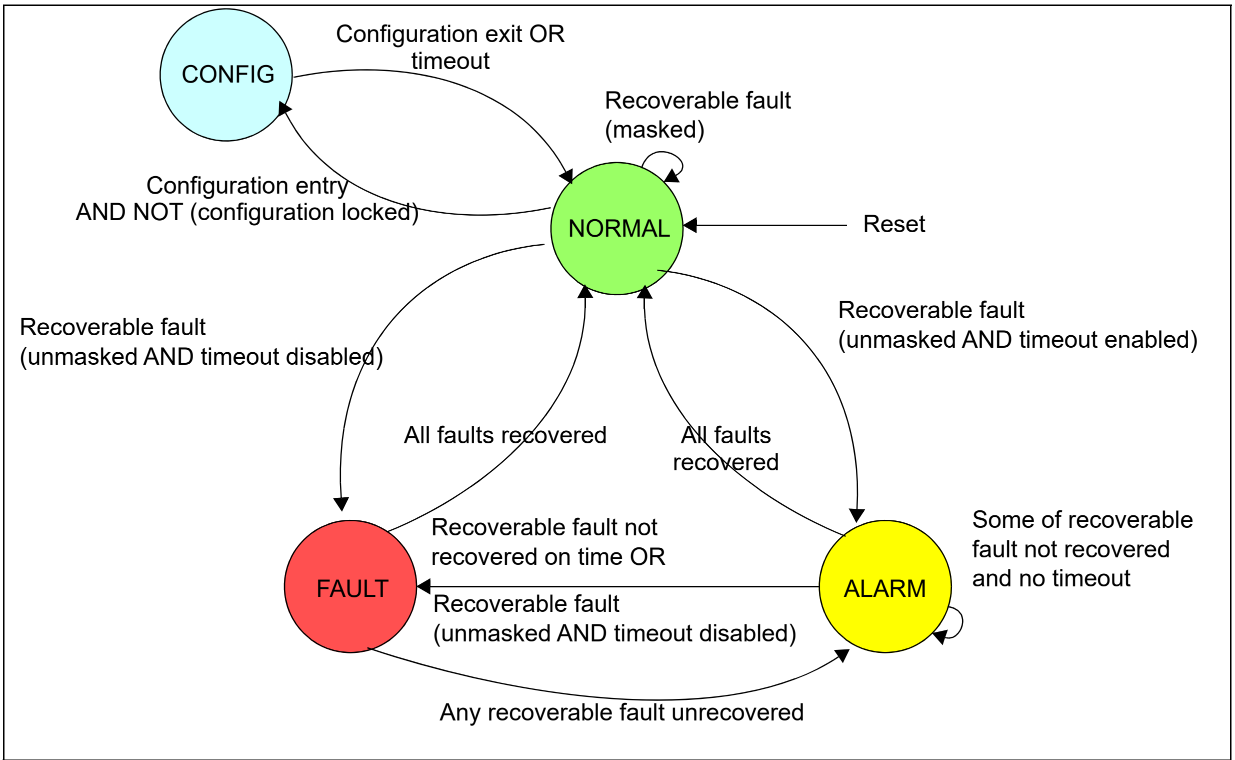
- FAULT: the FCCU moves into the FAULT state when one of the following conditions occurs:
  - timeout related to a recoverable fault when the FCCU is in the ALARM state.
  - unmasked recoverable faults with the timeout disabled.

The transition from NORMAL/ALARM state goes along with the generation of:

- NMI interrupt (optional).
- EOUT signaling (optional).
- SW option: Soft reaction (Short functional reset request pulse if configured).
- SW option: Hard reaction (Long functional reset request pulse if configured).

Non Maskable Interrupt (NMI) is routed only to the Safety Core.

Figure 1517. FCCU state diagram



75.5.3 Reset interface

The FCCU has two input resets, as described in *Table 1477* and two output resets related to the FAULT state. In FAULT state, each fault can be programmed to generate a soft or hard reaction request pulse.

In *Table 1478* for each type of fault reaction (hard, soft or self-checking), the respective output reset (Long functional reset, Short functional reset) is activated. In *Table 1479* for each type of reset source (external reset, POR, STCU, FCCU failure, short functional, long functional), the respective FCCU input reset state is described.

The MC unit generates the corresponding input reset as described in *Table 1477*.

Table 1477. Input reset

Input reset	Description
System reset	Global reset of the FCCU synchronous to the system clock.
System reset (RC oscillator)	Global reset of the FCCU synchronous to the IRCOSC.

Table 1478 and *Table 1479* describe a typical reset strategy at system level.

Table 1478. Reset reactions

FAULT reaction	Active reset	System reset signal
Hard	Long functional reset	disabled
Soft	Short functional reset	disabled

Table 1479 shows the Reset sources.

Table 1479. Reset sources

Reset source	System reset signal
External reset	enabled
POR	enabled
STCU	enabled
FCCU failure	application dependent
Functional reset	disabled

Table 1480 summarizes the MC actions corresponding to each reset generated by FCCU.

Table 1480. MC actions

Active reset	MC action
Long functional reset	Reset Flash and digital modules excluded FCCU, STCU
Short functional reset	Reset digital modules excluded FCCU, STCU

In reset state the FCCU is not able to collect any RF. The FCCU reset phase (POR, external reset, STCU, destructive reset, FCCU failure) must guarantee the assertion of the FCCU input resets at least for two IRCOSC clock periods.

75.5.4 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (possible for RFs only) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a RF (unmasked and with timeout disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the RF (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the RF (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU will return in the ALARM state (if any RF is still pending and the timeout is not elapsed).

In general, no fault nesting is supported except for the RF that causes an ALARM to FAULT state transition. In this case the alarm timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

75.5.5 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a RF management (ALARM state), refer to *Figure 1518* and *Figure 1519*, is as follows:

- RF assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - Alarm interrupt request (if configured)
  - Timeout running
- System state: RUN
- Alarm interrupt management: FAULT recovery (by SW): FCCU state transition ALARM → NORMAL

Figure 1518. RF (ALARM state) recovery (a)

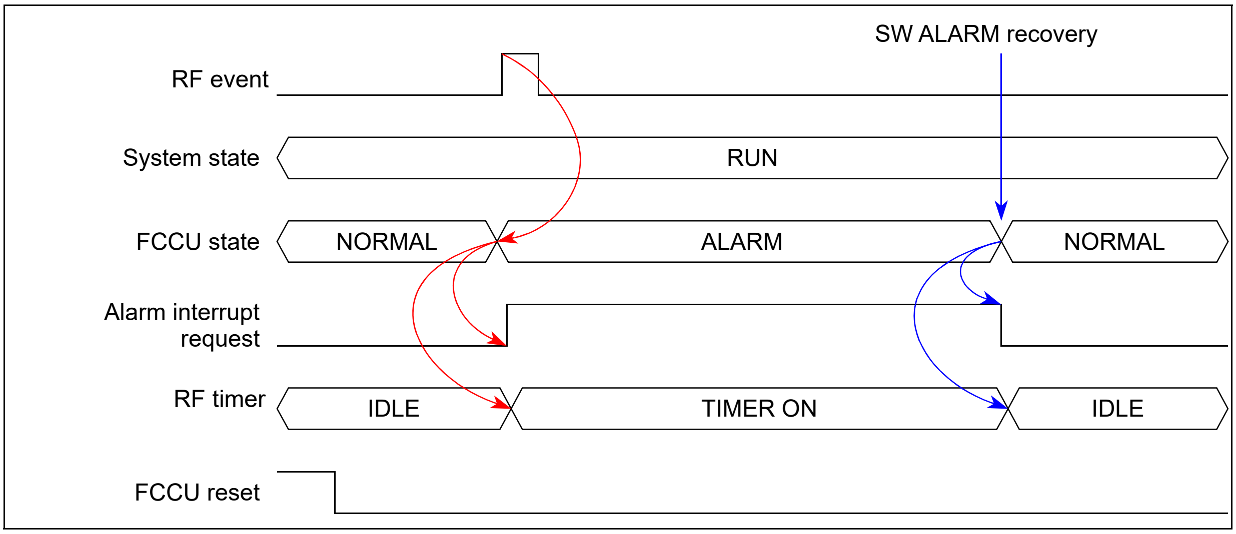
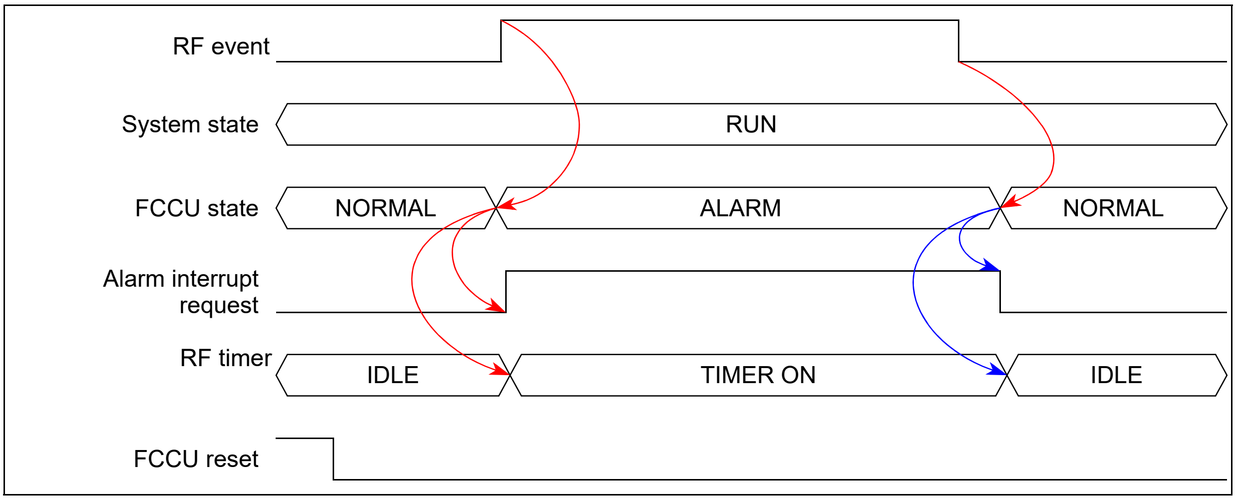


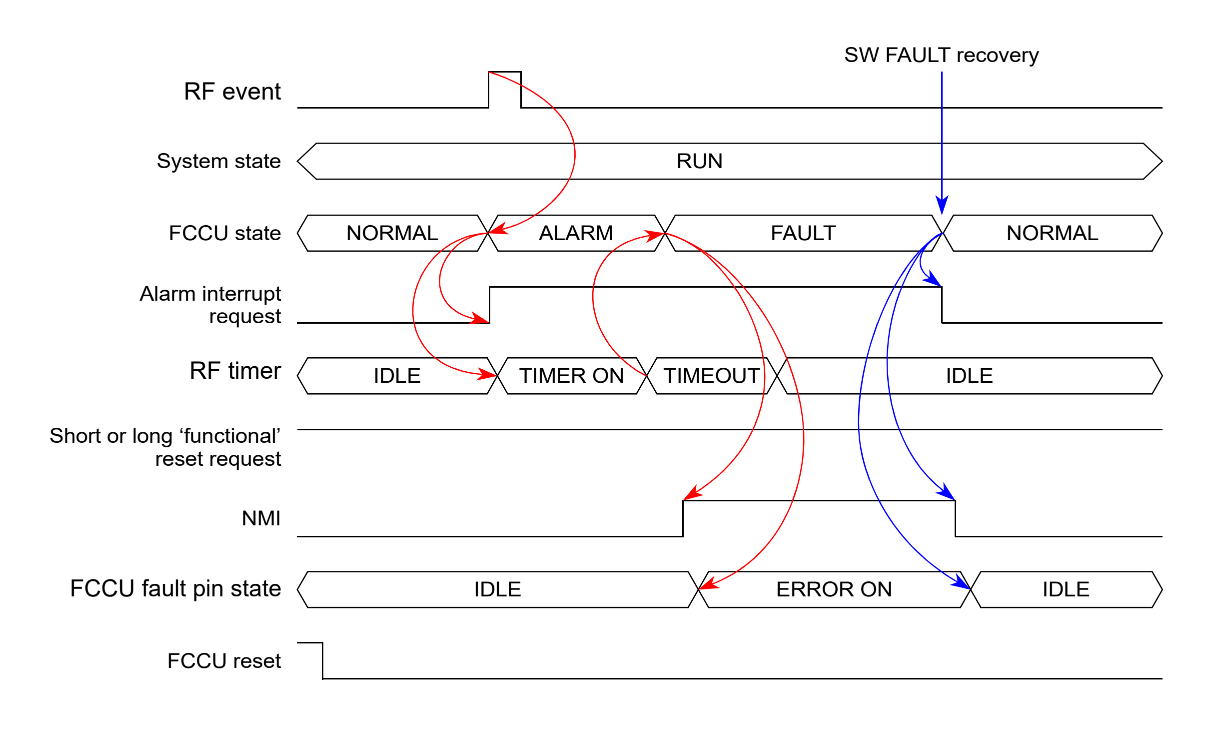
Figure 1519. RF (ALARM state) recovery (b)



A typical sequence related to a RF management (ALARM → FAULT state), refer to *Figure 1520*, is as follows:

- RF assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - Alarm interrupt request (if enabled)
  - Timeout running
- FCCU state transition (following the timeout trigger): ALARM → FAULT: NMI assertion (if enabled)
- NMI interrupt management (if enabled)
  - FAULT recovery (by SW): FCCU state transition FAULT → NORMAL

Figure 1520. RF(ALARM -> FAULT state) recovery



75.5.6 NMI/WKPU interface

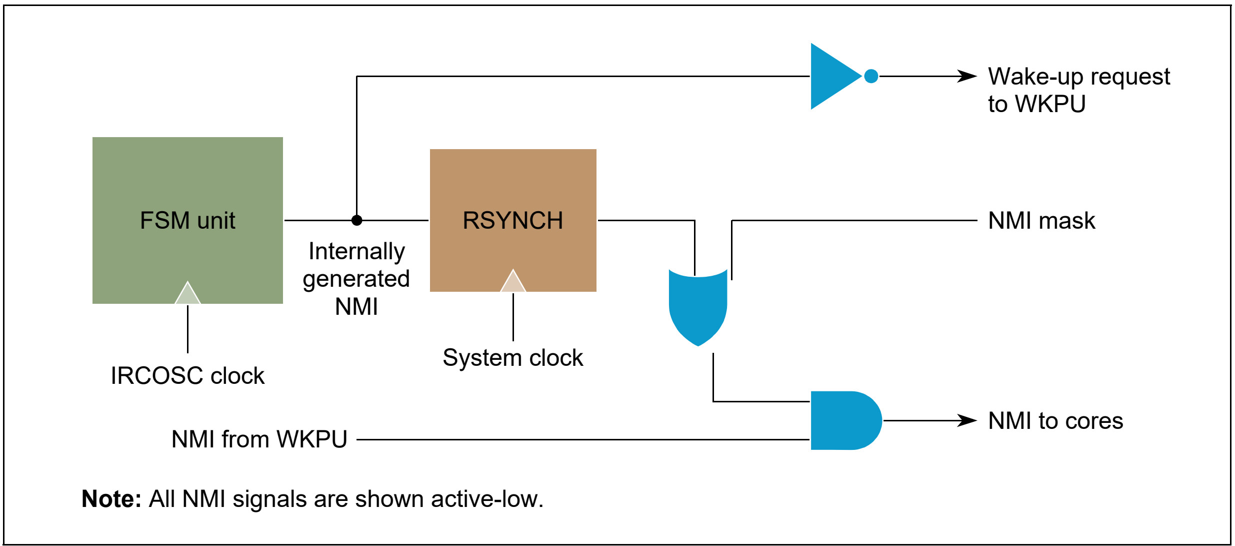
The wake-up interface includes Non-maskable interrupt: NMI interrupt source generated by the WKPU. It is synchronous with the system clock and active low. A logical AND interconnects the NMI sources (internally generated by FCCU and by WKPU) in order to provide a single NMI output.

The NMI signal internally generated by FCCU is masked when:

- FCCU asynchronous reset
- MC's current mode (MC state[3:0]) = RESET

and unmasked when a SW change request of the MC state is triggered (MC mode SW change request = 1).

Figure 1521. NMI/WKUP scheme



75.5.7 FAULT interface

The basic functionality of the fault interface is to provide a simple, programmable edge-sensitive interface (active high or low) to signal a fault condition to the FCCU module.

To support a fault injection mechanism an additional and optional signal is available. It is an active high signal and is asserted for a single clock cycle synchronously at the system clock. The behavior is given in *Figure 1522*. The fault injection is executed by a write operation in the FCCU_RFF register.The reaction following a fake RF cannot be masked. The fault injection mechanism is optional; refer to the implementation-specific details. Refer to “FCCU failure inputs” table in Functional Safety chapter, for more details.

To support a fault clearing mechanism, in order to clear a fault latched directly in the FAULT root, an additional and optional signal (FCCU recoverable faults clear) is available. This active-high signal is asserted for a single clock cycle synchronously at the system clock. The behavior is given in *Figure 1523*. The fault clearing is executed when the SW “clears” the status (flag) bits of the FCCU_RF_S*n*. The fault clearing mechanism is optional and its support must be specified at system level.

Figure 1522. FAULT injection

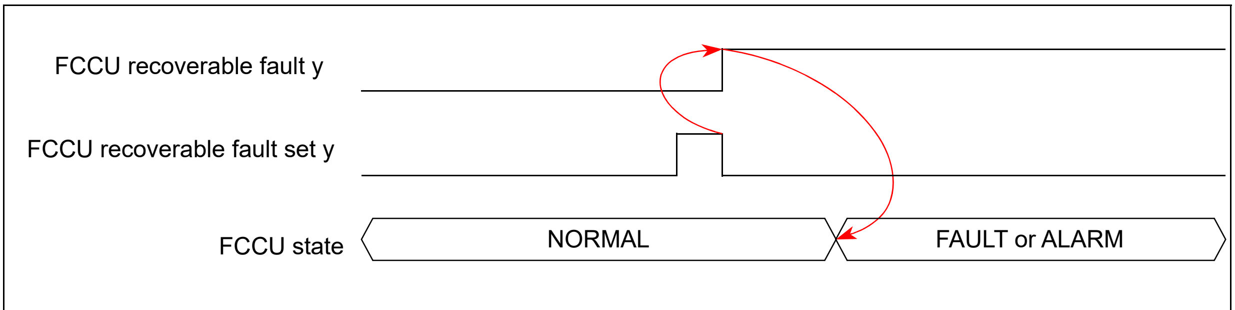
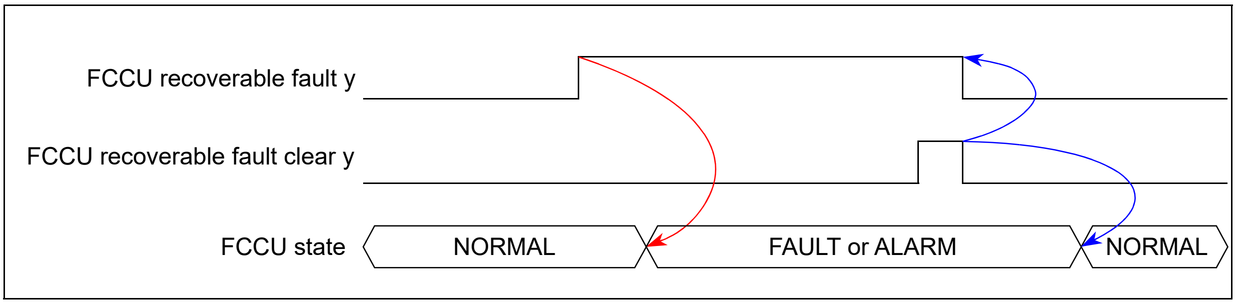


Figure 1523. FAULT clearing



75.5.8 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the following FCCU fault:
  – Recoverable
- The STCU fault signals are processed by the FCCU when the SOC is rebooted following the self-testing procedure. The STCU includes also a status register that stores the self-testing results (flags).
- A mask (EOUT mask[1:0]) that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self testing procedure, depending on the STCU results, three cases are applicable:

a) STCU completes the self testing procedure successfully. The SOC reboots and the FCCU is responsible for providing a reaction.

b) STCU completes the self testing procedure with low severity failures. The FCCU is responsible for providing the proper reactions according to the fault occurred.

c) STCU completes the self testing procedure with serious failures. The FCCU or other critical parts of the SoC could not be able to provide the proper reaction. STCU should keep permanently the SoC in RESET state. This feature is optionally programmable inside the STCU.

Figure 1524. STCU-FCCU (case a)

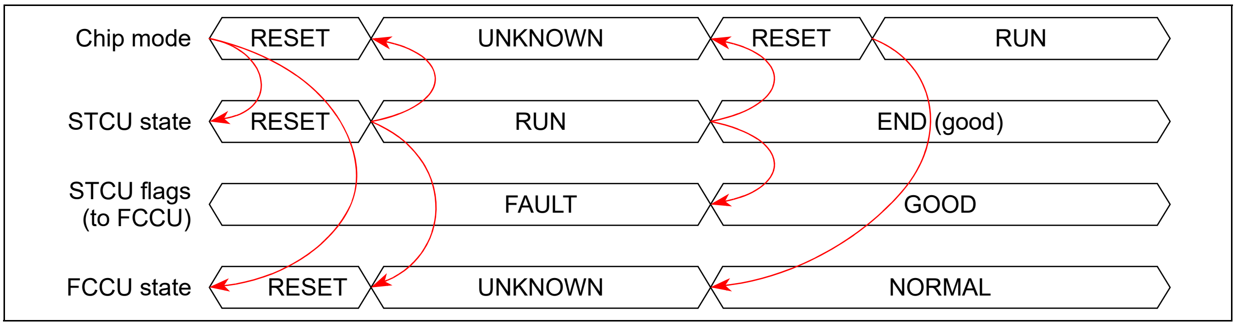


Figure 1525. STCU-FCCU (case b)

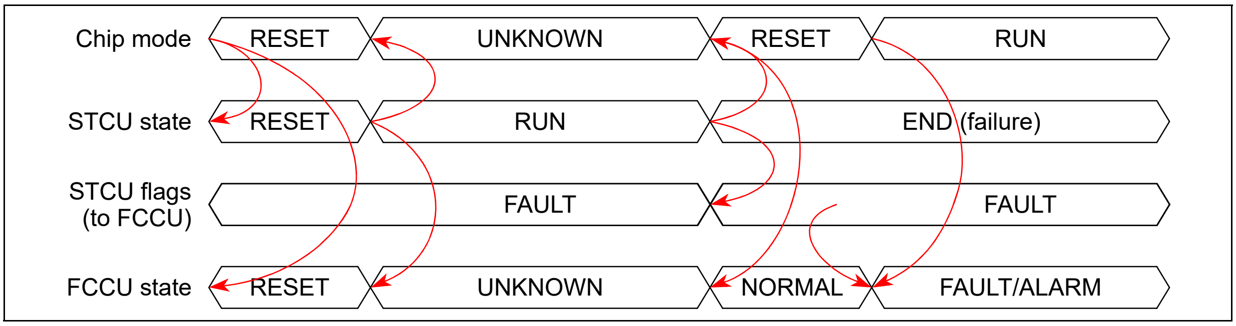
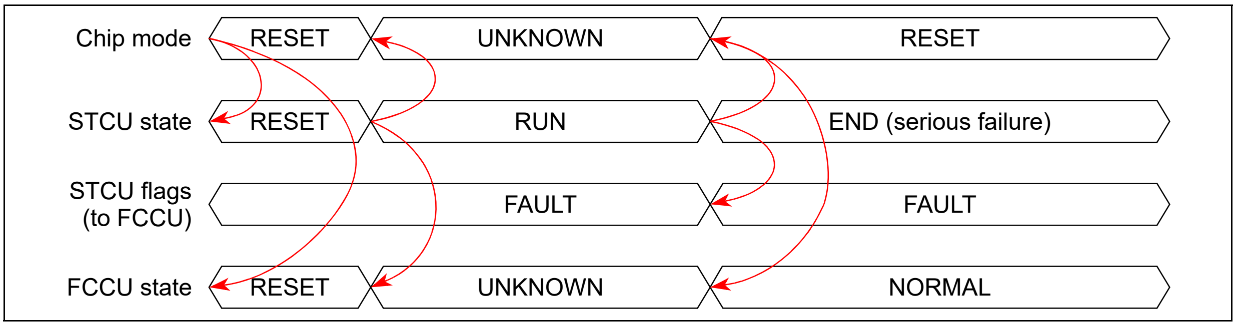


Figure 1526. STCU-FCCU (case c)



75.5.9 EOUT interface

The FCCU provides up to two bidirectional signals (EOUT interface) as a failure indication to the external world (refer to table “FCCU failure inputs” in the “Functional safety” chapter for details). These signals need to be mapped to appropriate pins for external visibility. Since SIUL2 does not control the pad buffer enable during Alternate function, FCCU provides a mechanism to drive the pad in OD (Open Drain) mode where the pad is driven Hi-Z when FCCU drives logic ‘1’ else it drives logic ‘0’ (when FCCU drives logic '0'). The selection to OD is done by OD bit in FCCU_CFG register. Different protocols for the EOUT interface are supported, selecting the FCCU_CFG.FOM register field:

- Dual rail protocol
- Time switching protocol
- Bi-stable protocol
- Test mode

The signal polarity can be programmed by setting the PS field in the FCCU_CFG register. For example, for PS = 0, FCCU Error output[0] = 0 and FCCU Error output[1] = 1 in FAULT state for bi-stable and for PS = 1, it is inverted. All the diagrams and tables are related to the default configuration selection: switching mode (FCCU_CFG.SM = 0b). In case of inverted polarity (FCCU_CFG.PS = 1b) all the values on the EOUT output pins are inverted.

Two modes can be programmed to define the EOUT protocol transitions in dual rail or time switching mode:

- slow switching mode: no EOUT frequency violation during the FCCU state transition (NORMAL to FAULT or vice-versa and CONFIG to NORMAL). The EOUT protocol transition occurs after a max delay equal to the duration of the semi-period of the EOUT frequency.

- fast switching mode: The EOUT protocol transition (NORMAL to FAULT or vice-versa and CONFIG to NORMAL) occurs immediately. A pulse with the minimum duration corresponding to 16 MHz/1024 (IRCOSC clock) period can occur in fast switching mode. It implies a frequency violation of the EOUT protocol.

The EOUT frequency is generated by dividing IRCOSC clock by a fixed factor of 2^{18}. With an IRCOSC clock of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and the IRCOSC clock detecting the edge transition of the EOUT protocol in dual rail or time switching mode.

In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before. If SW configures the error pins to OK(1) and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and error pins indicate NOK, that is an incoming fault is not masked when SW has set the error pin to high. During the T_min by a non-SW fault, the FCCU FSM moves independently of this pin state (low) and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in and it is not possible to set the pins to OK by SW moving FCCU to CONFIG state, as long as this timer is running. No SW intervention is needed to bring the pin from the low state. The SW can bring the pin back to OK state by clearing the faults and waiting for the T_min to expire, after which the FCCU automatically enters NORMAL state and the error pin indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

Resets (except the power on reset, and software bit FCCU_RST in RGM_PRST5) should not influence the state of the failure indication, pins.

Note: The transition from FAULT to CONFIG state is not possible but is shown to display the behavior in all four states: RESET, NORMAL, FAULT and CONFIG.

Table 1481. FCCU error pin behavior on STATE transitions(1)

Fault signaling state	Transition from NORMAL to FAULT state			NORMAL+CONFIG states			Transition from NORMAL to ALARM state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal Error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

1. Table valid only after SW has written FCCU_SET_AFTER_RESET bit to 1.

Note: FCCU remains in NORMAL state when fault is disabled hence, “NA” (Not Applicable) appears in the above table.

75.5.9.1 Dual rail protocol

Dual rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual rail encoded circuits use two wires to carry each bit. The encoding scheme is given in Table 1482 and the related timing diagram is given in Figure 1527 and Figure 1528.

Table 1482. Dual rail encoding

Logical state	Dual rail encoding (FCCU Error output pins [1:0])	Note
nonfaulty	10	toggling
nonfaulty	01	
faulty	00	toggling
faulty	11	
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	toggling

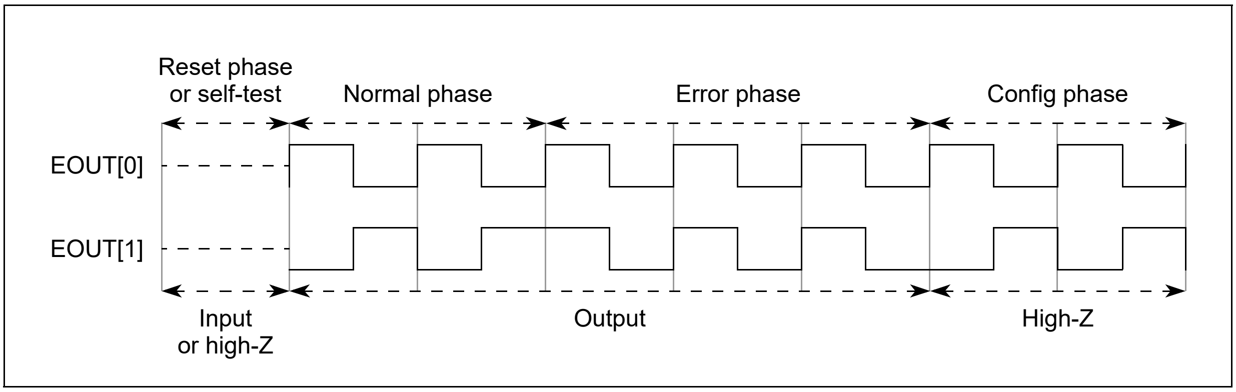
1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

As long as FCCU is in NORMAL or ALARM state, output will show “nonfaulty” signal. Output pins #0, #1 will toggle between 01 and 10.

In the RESET phase the output pins are set as “high impedance”[v].

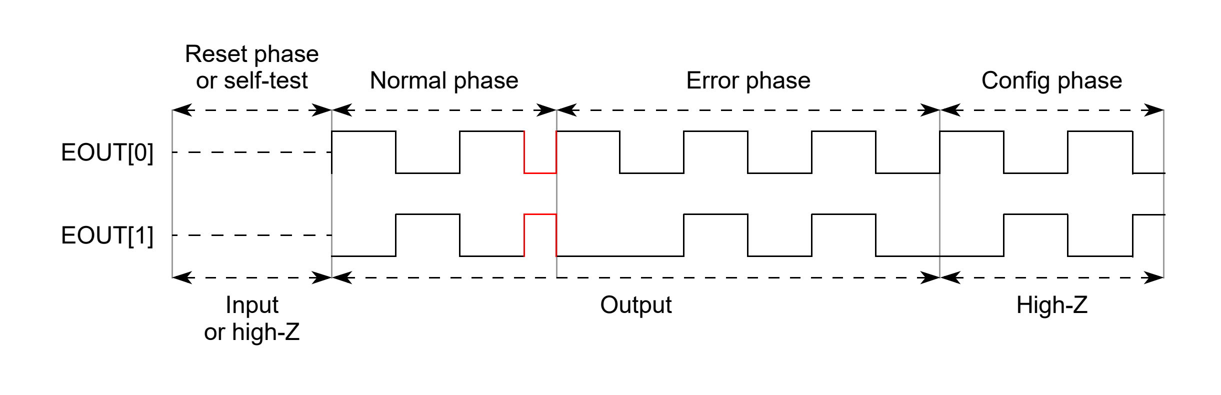
Note: Figure 1527 and Figure 1528 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

Figure 1527. Dual rail protocol (slow switching mode)



v. Actual value depends on the SOC settings at pad level.

Figure 1528. Dual rail protocol (fast switching mode)



75.5.9.2 Time switching protocol

The encoding scheme is given in *Table 1483* and the related timing diagram is given in *Figure 1529*.

Table 1483. Time switching encoding

Logical state	Time switching encoding (FCCU Error output pins [1:0])	Note
nonfaulty	10	toggling
nonfaulty	01	
faulty	10	no toggling
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	toggling

1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

As long as FCCU is in NORMAL or ALARM state, outputs will show “nonfaulty” signal. Output pins #0, #1 will toggle between 01 and 10.

In the FAULT state, the output pin FCCU Error output pin[0] is set as low.

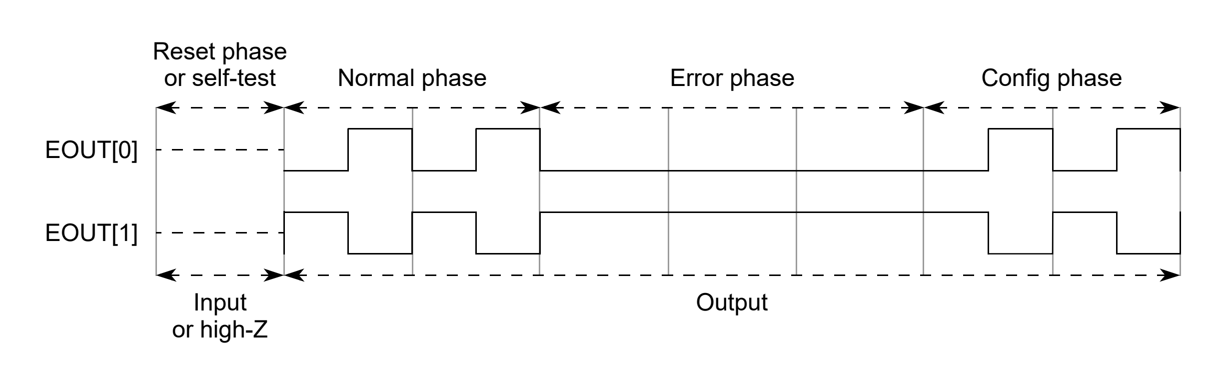
In Time Switching mode the second output (FCCU Error output pin[1]) is the inverted signal of first output (FCCU Error output pin[0]).

In the RESET phase the output pins are set as “high impedance”[w].

Note: Figure 1529 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

w. Actual value depends on the SOC settings at pad level.

Figure 1529. Time switching protocol



75.5.9.3 Bi-stable protocol

The encoding scheme is given in Table 1484 and the related timing diagram is given in Figure 1530.

Table 1484. Bi-stable encoding

Logical state	Bi-stable encoding (FCCU Error output pins [1:0])	Note
nonfaulty	01	no toggling
faulty	10	no toggling
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	no toggling

1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

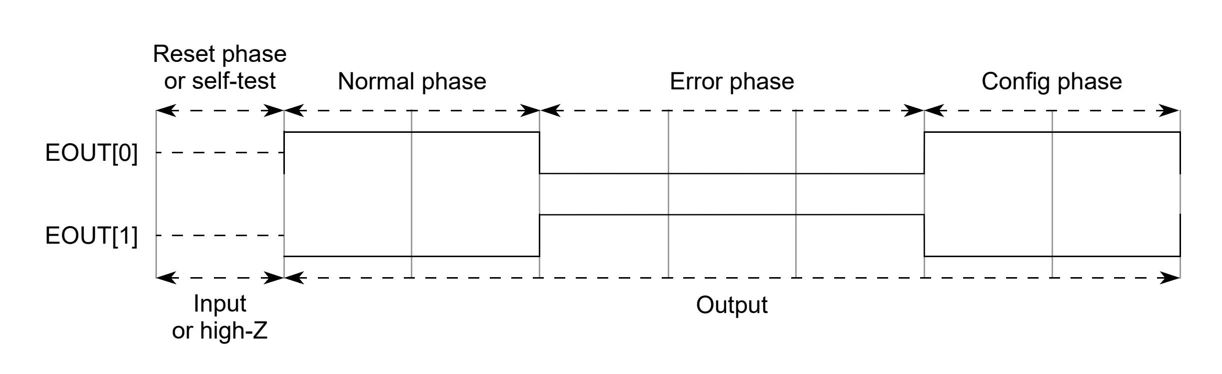
In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, "no-faulty" state is indicated. In Bi-stable mode the second output (FCCU Error output[1]) is the inverted signal of first output (FCCU Error output[0]).

In the RESET phase the output pins are set as "high impedance"(x).

Note: Figure 1530 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

x. Actual value depends on the SOC settings at pad level.

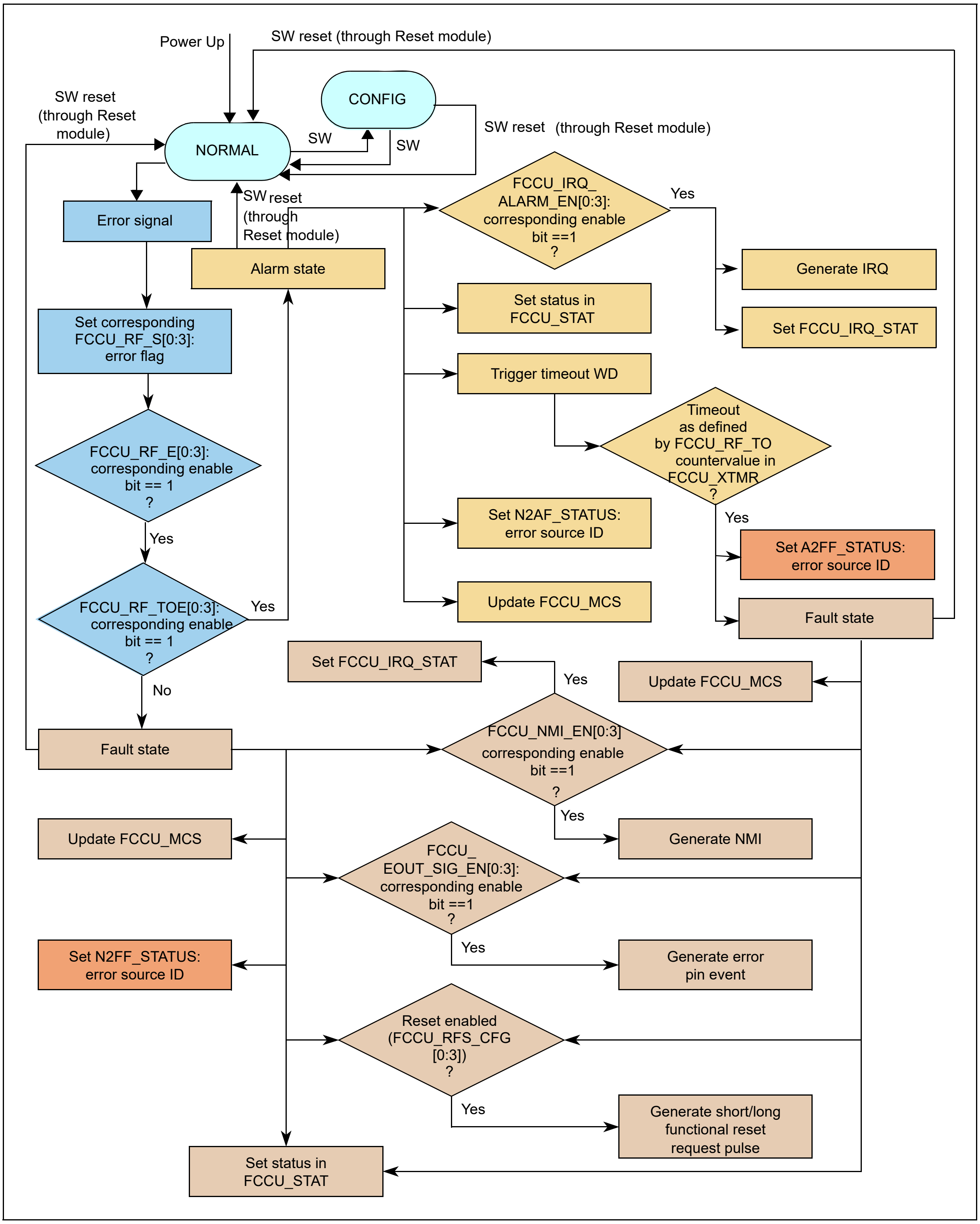
Figure 1530. Bi-stable protocol



75.5.10 Error signal flow for RF case

On Power up, FCCU moves to NORMAL state. SW can configure it by moving it to CONFIG and back again. SW reset leads to movement back to NORMAL. On reaching ALARM state, depending on ALARM being enabled, FCCU generates IRQ (alarm) and the FCCU_IRQ_STAT's alarm status is also set. FSM status is set in FCCU_STAT along with N2AF_STATUS. If Error is enabled (FCCU_RF_E) and timeout disabled (FCCU_RF_TOE), FCCU moves straight to FAULT state and depending on FCCU_NMI_EN or FCCU_SIG_EN, or both, being set, it generates a NMI or signals error, or both, out to the external world. FCCU FSM status is also captured in FCCU_STAT.

Figure 1531. Error signal flow



75.6 Register description

The FCCU registers are listed in *Table 1485*. Any address offset not explicitly mentioned in *Table 1485* is reserved.

All the registers accessible in write mode only in CONFIG state (last column of the *Table 1485*) are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible RF failure source a different reaction shall be configurable through the use of NMI, IRQ, long/short reset selection registers as well as no reaction by disabling the former registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

Table 1485. FCCU memory map

Address offset	Register	Access(1)	Location
0x000	FCCU Control Register (FCCU_CTRL)	R/W always	*Section 75.6.1*
0x004	FCCU CTRL Key Register (FCCU_CTRLK)	W always	*Section 75.6.2*
0x008	FCCU Configuration Register (FCCU_CFG)	R always; W in CONFIG state only	*Section 75.6.3*
0x01C	FCCU RF Configuration Register 0 (FCCU_RF_CFG0)	R always; W in CONFIG state only	*Section 75.6.4*
0x020	FCCU RF Configuration Register 1 (FCCU_RF_CFG1)		
0x024	FCCU RF Configuration Register 2 (FCCU_RF_CFG2)		
0x028	FCCU RF Configuration Register 3 (FCCU_RF_CFG3)		
0x04C	FCCU RFS Configuration Register 0 (FCCU_RFS_CFG0)	R always; W in CONFIG state only	*Section 75.6.5*
0x050	FCCU RFS Configuration Register 1 (FCCU_RFS_CFG1)		
0x054	FCCU RFS Configuration Register 2 (FCCU_RFS_CFG2)		
0x058	FCCU RFS Configuration Register 3 (FCCU_RFS_CFG3)		
0x05C	FCCU RFS Configuration Register 4 (FCCU_RFS_CFG4)		
0x060	FCCU RFS Configuration Register 5 (FCCU_RFS_CFG5)		
0x064	FCCU RFS Configuration Register 6 (FCCU_RFS_CFG6)		
0x068	FCCU RFS Configuration Register 7 (FCCU_RFS_CFG7)		
0x080	FCCU RF Status Register 0 (FCCU_RF_S0)	R/W always	*Section 75.6.6*
0x084	FCCU RF Status Register 1 (FCCU_RF_S1)		
0x088	FCCU RF Status Register 2 (FCCU_RF_S2)		
0x08C	FCCU RF Status Register 3 (FCCU_RF_S3)		

Table 1485. FCCU memory map (continued)

Address offset	Register	Access(1)	Location
0x090	FCCU RF Key Register (FCCU_RFK)	W always	Section 75.6.7
0x094	FCCU RF Enable Register 0 (FCCU_RF_E0)	R always; W in CONFIG state only	Section 75.6.8
0x098	FCCU RF Enable Register 1 (FCCU_RF_E1)		
0x09C	FCCU RF Enable Register 2 (FCCU_RF_E2)		
0x0A0	FCCU RF Enable Register 3 (FCCU_RF_E3)		
0x0A4	FCCU RF Timeout Enable Register 0 (FCCU_RF_TOE0)	R always; W in CONFIG state only	Section 75.6.9
0x0A8	FCCU RF Timeout Enable Register 1 (FCCU_RF_TOE1)		
0x0AC	FCCU RF Timeout Enable Register 2 (FCCU_RF_TOE2)		
0x0B0	FCCU RF Timeout Enable Register 3 (FCCU_RF_TOE3)		
0x0B4	FCCU RF Timeout Register (FCCU_RF_TO)	R always; W in CONFIG state only	Section 75.6.10
0x0B8	FCCU CFG Timeout Register (FCCU_CFG_TO)	R always; W in all states except CONFIG	Section 75.6.11
0x0BC	FCCU IO Control Register (FCCU_EINOUT)	R/W always	Section 75.6.12
0x0C0	FCCU Status Register (FCCU_STAT)	R only	Section 75.6.13
0x0C4	FCCU NA Freeze Status Register (N2AF_STATUS)	R/W always	Section 75.6.14
0x0C8	FCCU AF Freeze Status Register (A2FF_STATUS)	R/W always	Section 75.6.15
0x0CC	FCCU NF Freeze Status Register (N2FF_STATUS)	R/W always	Section 75.6.16
0x0D0	FCCU FA Freeze Status Register (F2A_STATUS)	R/W always	Section 75.6.17
0x0DC	FCCU RF Fake Register (FCCU_RFF)	R/W always	Section 75.6.18
0x0E0	FCCU IRQ Status Register (FCCU_IRQ_STAT)	R/W always	Section 75.6.19
0x0E4	FCCU IRQ Enable Register (FCCU_IRQ_EN)	R/W always	Section 75.6.20
0x0E8	FCCU XTMR Register (FCCU_XTMR)	R/W always	Section 75.6.21
0x0EC	FCCU MCS Register (FCCU_MCS)	R only	Section 75.6.22
0x0F0	FCCU Transient Lock Register (FCCU_TRANS_LOCK)	W always	Section 75.6.23
0x0F4	FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)	W always	Section 75.6.24
0x0F8	FCCU Delta T Register (FCCU_DELTA_T)	R always; W in CONFIG state only	Section 75.6.25

Table 1485. FCCU memory map (continued)

Address offset	Register	Access[1]	Location
0x0FC	FCCU IRQ Alarm Enable Register 0 (FCCU_IRQ_ALARM_EN0)	R always; W in CONFIG state only	Section 75.6.26
0x100	FCCU IRQ Alarm Enable Register 1 (FCCU_IRQ_ALARM_EN1)		
0x104	FCCU IRQ Alarm Enable Register 2 (FCCU_IRQ_ALARM_EN2)		
0x108	FCCU IRQ Alarm Enable Register 3 (FCCU_IRQ_ALARM_EN3)		
0x10C	FCCU NMI Enable Register 0 (FCCU_NMI_EN0)	R always; W in CONFIG state only	Section 75.6.27
0x110	FCCU NMI Enable Register 1 (FCCU_NMI_EN1)		
0x114	FCCU NMI Enable Register 2 (FCCU_NMI_EN2)		
0x118	FCCU NMI Enable Register 3 (FCCU_NMI_EN3)		
0x11C	FCCU EOUT Signaling Enable Register 0 (FCCU_EOUT_SIG_EN0)	R always; W in CONFIG state only	Section 75.6.28
0x120	FCCU EOUT Signaling Enable Register 1 (FCCU_EOUT_SIG_EN1)		
0x124	FCCU EOUT Signaling Enable Register 2 (FCCU_EOUT_SIG_EN2)		
0x128	FCCU EOUT Signaling Enable Register 3 (FCCU_EOUT_SIG_EN3)		

1. In this column, R/W = read/write, R = read-only, and W = write-only.

75.6.1 FCCU Control Register (FCCU_CTRL)

The FCCU_CTRL register allows to execute the following operations:

- moving the FCCU state from the NORMAL state into the CONFIG state.
- moving the FCCU state from the CONFIG state into the NORMAL state.
- reading or to clear the RF status register.
- reading the FCCU FSM status register.
- reading or to clear the FCCU freeze registers.
- reading the ALARM timer.
- reading the Watchdog timer.
- configure the glitch filter present on EIN0 input (FCCU failure input #96).

Some critical operations require a key as defined in the FCCU_CTRLK register.

The Glitch filter, filters out pulses of widths up to 50/75/100 µs, as governed by FCCU_CTRL[FILTER_WIDTH]. It becomes active as soon as FCCU_CTRL[FILTER_BYPASS] is de-asserted (default value).

The implementation is as follows:

1. Fault line assertion - If a Fault is asserted, then the output of the filter is asserted after the delay, equal to the programmed glitch width time. If the fault is de-asserted before

the programmed glitch width time, then the output of the filter is not asserted and stays at the deactivated level (after having rejected the spurious glitch).

2. Fault line de-assertion - If a Fault, which was kept asserted for a time greater than the programmed pulse width is de-asserted (and glitch filter output is at asserted level now), then the output of the filter is de-asserted only after the delay equal to the programmed glitch width time. If the fault is again asserted before the programmed glitch width time, then the output of the filter is not de-asserted and stays at the active level (after having rejected the spurious glitch).

The output of the filter becomes the effective fault input fed to the FCCU error management engine. Since there is an inherent latency involved, SW must either wait for the programmed delay value before trying to clear the fault status flag, FCCU failure input #96 or first set FCCU_CTRL[FILTER_BYPASS] to one and then try to clear the fault status flag. This bit should not be made '1' again, till the filter width time elapses, else fault status will be latched again.

Offset: 0x000 Access: User read/write

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FILTER_BYPASS	FILTER_WIDTH		0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	DEBUG	0	OPS		0	OPR				
W																
Reset	0	0	0	0	0	0	0	0	0/1(1)	0/1(1)	0	0	0	0	0	0

1. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 1532. FCCU Control Register (FCCU_CTRL)

Table 1486. FCCU_CTRL field descriptions

Field	Description
0 FILTER_BYPASS	Filter bypass 0 glitch filter not bypassed 1 glitch filter bypassed
1:2 FILTER_WIDTH	Filter width 00 filters glitches up to 50 µs 01 filters glitches up to 75 µs 10 filters glitches up to 100 µs for 20 MHz IRC clock 11 filters glitches up to 100 µs for 20 MHz IRC clock Make sure that the filter width does not cross the FOSU timeout value else unintended reset may occur.

Table 1486. FCCU_CTRL field descriptions (continued)

Field	Description
22 DEBUG	Debug mode entry 0 Normal operation 1 Put FCCU into debug mode When Debug pin signal is asserted and this bit is set, FCCU moves into debug state and suspend the transition of the FSM by suspending timer elapse time effect.
24:25 OPS	Operation status 00 Idle 01 In progress 10 Aborted 11 Successful These bits can be read and cleared (via OP15 operation) by the software.

Table 1486. FCCU_CTRL field descriptions (continued)

Field	Description
27:31 OPR	Operation run 00000 No operation [OP0] 00001 Set the FCCU into the CONFIG state [OP1] 00010 Set the FCCU into the NORMAL state [OP2] 00011 Read the FCCU state (refer to the FCCU_STAT register) [OP3] 00100 Read the FCCU frozen status flags (refer to the N2AF_STATUS register) [OP4] 00101 Read the FCCU frozen status flags (refer to the A2FF_STATUS register) [OP5] 00110 Read the FCCU frozen status flags (refer to the N2FF_STATUS register) [OP6] 00111 Read the FCCU frozen status flags (refer to the F2A_STATUS register) [OP7] 01000 Reserved 01001 Reserved 01010 Read the RF status register (refer to the FCCU_RF_S register) [OP10] 01011 Reserved 01100 RF status clear operation in progress (refer to the FCCU_RF_S register) [OP12] 01101 Clear the freeze status registers (refer to the freeze registers) [OP13] 01110 CONFIG to NORMAL FCCU state (configuration timeout) in progress [OP14] 01111 Clear the operation status (OPS=Idle) [OP15] 10000 Reserved 10001 Read the ALARM timer (refer to the FCCU_XTMR register) [OP17] 10010 RESERVED 10011 Read the CFG timer (refer to the FCCU_XTMR register) [OP19] 10100 Read the Error Pin low counter value (refer to *Section 75.6.21: FCCU XTMR Register (FCCU_XTMR)*) [OP20] 10101 Reserved ... 11111 Reserved The SW application must not modify the OPR field (any write operation will be ignored) until the completion of the operation. Once the operation has been completed, the operation status (OPS) is set and the OPR field is automatically cleared (OPR = 000). The opcode OP11, OP12 must not be programmed. The OPR field is automatically set to OP11 or OP12 when the FCCU_RF_S registers are cleared by a write-clear operation into the related register. The opcode OP14 must not be programmed. The OPR field is automatically set to OP14 when the timeout occurs (FCCU_CFG_TO) during the configuration procedure => the FCCU state is automatically forced in NORMAL mode setting the default configuration. In this phase any write operation to the FCCU configuration registers is inhibited. The reserved operations (OP21 -OP30) are forbidden. In case of execution, they return an ABORT response without any side effect. The ABORT response occurs in the following cases: – wrong access (missing or wrong key) to the FCCU_RF_S register (clear operation OP12) – wrong access (missing or wrong key) to the FCCU_CTRL register (OP1, OP2 operation) – OP1 (CONFIG command) execution when FCCU state /= NORMAL or configuration locked – OP21-OP30 (RESERVED operations) execution The max timing needed for ABORT/SUCCESS response is 15 IRC cycles

75.6.2 FCCU CTRL Key Register (FCCU_CTRLK)

The FCCU_CTRLK register implements the key access for the operations OP1, OP2 according to the following sequence:

1. Write the key into the FCCU_CTRLK register.
2. Write the FCCU_CTRL register (operations OP1 or OP2).

Note: There shall not be any other operation in between the above steps and the 32 bits need to be written in one cycle. So, store FCCU_CTRLK, merge with OP1 or OP2 then send it.

The FCCU_CTRLK register is not readable, a 0x0000_0000 value is always returned in case of read operation. The key must be written by a word (32 bits) data write operation.

Offset: 0x004 Access: User write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	CTRLK_VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	CTRLK_VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1533. FCCU CTRL Key Register (FCCU_CTRLK)

Table 1487. FCCU_CTRLK field descriptions

Field	Description
0:31 CTRLK_VAL	Control register key 0x913756AF Key for the operation OP1 0x825A132B Key for the operation OP2

75.6.3 FCCU Configuration Register (FCCU_CFG)

The FCCU_CFG register defines the global configuration for the FCCU module.

This register is writable only in the CONFIG state.

Offset: 0x008 Access: User read/write[(1)]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							FCCU_SET_AFTER_RESET	FCCU_SET_CLEAR		0	0	Reserved			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	Reserved		OD	0	SM	PS	FOM			Reserved					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0[(2)]	0	0	0

1. Writable only in the CONFIG state.
2. After powerup, reset value will be 0. If configuration timeout occurs then default value will be changed to 1.

Figure 1534. FCCU Configuration Register (FCCU_CFG)

Table 1488. FCCU_CFG field descriptions

Field	Description
7 FCCU_SET_ AFTER_RESET	This bit controls the enable of the o/p error pin after reset lifts 0 FCCU's Error indication stops functioning and error pins are in HI-Z state[(1)]. 1 write to start FCCU functioning. After power-on the error out pins shall be in high impedance[(1)]. They will go to normal state only on software request, this bit is set to 1. It is Software responsibility to write this bit to 1 else the error pin stays in High-Z state after reset lifts and FCCU is not functioning.
8:9 FCCU_SET_ CLEAR	Error pin state can be controlled by these bits 01 Error Pin driven to logic 0. 11 Error Pin driven to logic 1 (write not possible when FCCU already in the T min timer phase or if FSM enters in FAULT state by capture of a fault). 00 FCCU acts independent of above SW control. 10 FCCU acts independent of above SW control. These bits clear(0) and set(1) the error pin. Higher priority is of the FCCU_SET_AFTER_RESET bit's capability to lead the error pins to Hi-Z(1). Switch to software configuration (01, 11) comes into effect when FCCU FSM leaves CONFIG state. After this is in effect, software maintains priority over FCCU FSM except for the case of '11' mentioned above. The switch back to FSM control is again done by writing into these fields with 00 or 10 by going to CONFIG state. This bit feature may not work with toggling protocols for example say EOUT pins are set high by SW using this bit followed by switching to a toggling protocol after returning control to FCCU FSM via setting this bit to 00 or 10, the user may find that EOUT pins do not toggle as was being expected.

Table 1488. FCCU_CFG field descriptions (continued)

Field	Description
19 OD	Open Drain Mechanism to select between Push-pull and Open drain (OD) mode for the error indicating pin(s). 0 push-pull 1 OD
21 SM	Switching mode 0 EOUT protocol (dual rail, time switching) slow switching mode. 1 EOUT protocol (dual rail, time switching) fast switching mode. SM has no effect on the bi-stable protocol.
22 PS	Polarity selection 0 FCCU Error output[1] active high, FCCU Error output[0] active low. 1 FCCU Error output[1] active low, FCCU Error output[0] active high. This bit can be read and written by the software. Active state here means a state indicating FAULT and applicable when pins are non toggling in Error phase.
23:25 FOM	Fault Output Mode selection 000 Dual Rail (default state) [FCCU Error output[1:0]= outputs] 001 Time Switching [FCCU Error output[1:0]= output to be used] 010 Bi-Stable 011 Reserved 100 Reserved 101 Test0 (controlled by FCCU_EOUT register) [FCCU Error output[0]= input, FCCU Error output[1]= output] 110 Test1 (controlled by FCCU_EOUT register) [FCCU Error output[0]= output, FCCU Error output[1]= output] 111 Test2 (controlled by FCCU_EOUT register) [FCCU Error output[0]= output, FCCU Error output[1]= input] **Note:** In Test mode, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs on the system/IRCOSC clock.

1. Actual value depends on the SOC settings at pad level.

75.6.4 FCCU RF Configuration Register *n* (FCCU_RF_CFG*n*)

The FCCU_RF_CFG*n* registers contain the configuration of each recoverable fault in terms of fault recovery management. The configuration depends on the type of signaling of a fault event. HW recoverable faults should be configured only if a previous latching stage captures and hold the physical fault otherwise the fault can be lost. All the other faults should be configured as SW fault.

These registers are writable only in the CONFIG state.

Offset: 0x01C + *n**0x4 (*n* = 0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	RFC31(2)	RFC30(2)	RFC29(2)	RFC28(2)	RFC27(2)	RFC26(2)	RFC25(2)	RFC24(2)	RFC23(2)	RFC22(2)	RFC21(2)	RFC20(2)	RFC19(2)	RFC18(2)	RFC17(2)	RFC16(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	RFC15(2)	RFC14(2)	RFC13(2)	RFC12(2)	RFC11(2)	RFC10(2)	RFC9(2)	RFC8(2)	RFC7(2)	RFC6(2)	RFC5(2)	RFC4(2)	RFC3(2)	RFC2(2)	RFC1(2)	RFC0(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 1490* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 1535. FCCU RF Configuration Register *n* (FCCU_RF_CFG*n*)

Table 1489. FCCU_RF_CFG*n* field descriptions

Field	Description
0:31 RFC[31:0]	Recoverable fault configuration 0 HW recoverable fault 1 SW recoverable fault The recoverable fault configuration defines the fault recovery mode. HW recoverable faults are self recovered (status flag clearing and related) if the root cause (input fault) has been removed. SW recoverable faults are recovered (status flag clearing) by SW clearing of the related status flag.

Table 1490: FCCU RF configuration register channels shows FCCU RF configuration register channels.

Table 1490. FCCU RF configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x01C	FCCU_RF_CFG0	RFC[31:0]
0x020	FCCU_RF_CFG1	RFC[63:32]
0x024	FCCU_RF_CFG2	RFC[95:64]
0x028	FCCU_RF_CFG3	RFC[127:96]

75.6.5 FCCU RFS Configuration Register *n* (FCCU_RFS_CFG*n*)

The FCCU_RFS_CFG*n* registers contain the configuration of each recoverable fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

These registers are writable only in the CONFIG state.

Offset: 0x04C +*n**0x4 (*n* = 0 to 7) Access: User read/write(1)

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	RFSC15(2)		RFSC14(2)		RFSC13(2)		RFSC12(2)		RFSC11(2)		RFSC10(2)		RFSC9(2)		RFSC8(2)	
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	RFSC7(2)		RFSC6(2)		RFSC5(2)		RFSC4(2)		RFSC3(2)		RFSC2(2)		RFSC1(2)		RFSC0(2)	
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 1492* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 1536. FCCU RFS Configuration Register *n* (FCCU_RFS_CFG*n*)

Table 1491. FCCU_RFS_CFG*n* field descriptions

Field	Description
0:31 RFSC[15:0]	Recoverable fault state configuration 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

Table 1492: FCCU RFS configuration register channels shows FCCU **RFS** configuration register channels.

Table 1492. FCCU RFS configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x04C	FCCU_RFS_CFG0	RFSC[15:0]
0x050	FCCU_RFS_CFG1	RFSC[31:16]
0x054	FCCU_RFS_CFG2	RFSC[47:32]
0x058	FCCU_RFS_CFG3	RFSC[63:48]
0x05C	FCCU_RFS_CFG4	RFSC[79:64]
0x060	FCCU_RFS_CFG5	RFSC[95:80]
0x064	FCCU_RFS_CFG6	RFSC[111:96]
0x068	FCCU_RFS_CFG7	RFSC[127:112]

75.6.6 FCCU RF Status register *n* (FCCU_RF_S*n*)

The FCCU_RF_S*n* register contains the latched fault indication collected from the recoverable fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the RF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_RF_E*n* register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_RF_E*n* register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_RF_S*n* register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_RF_S*n* register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_RFK register
2. Clear the status (flag) bit RFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field
3. Wait for the completion of the operation (FCCU_CTRL.OPS field)
4. Read the FCCU_RF_S*n* register in order to verify the effective deletion and in case of failure to repeat the sequence

As a result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root (refer to *Section 75.5.7: FAULT interface*).

Note: There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_RF_S*n* read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU_RF_S*n* register

In case of reconfiguration of the FCCU (CONFIG state), before returning in NORMAL state the pending status bits into the FCCU_RF_S*n* must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following errors are ignored:

- to write a wrong key into the FCCU_RFK register
- to attempt to clear a HW recoverable error

Offset: 0x080 + *n**0x4 (*n* = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFS31[1]	RFS30[1]	RFS29[1]	RFS28[1]	RFS27[1]	RFS26[1]	RFS25[1]	RFS24[1]	RFS23[1]	RFS22[1]	RFS21[1]	RFS20[1]	RFS19[1]	RFS18[1]	RFS17[1]	RFS16[1]
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset:	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFS15[1]	RFS14[1]	RFS13[1]	RFS12[1]	RFS11[1]	RFS10[1]	RFS9[1]	RFS8[1]	RFS7[1]	RFS6[1]	RFS5[1]	RFS4[1]	RFS3[1]	RFS2[1]	RFS1[1]	RFS0[1]
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset:	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]	0/1[2]

1. Refer to *Table 1494* for register offset to channel number relationship.
2. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 1537. FCCU RF Status register *n* (FCCU_RF_S*n*)

Table 1493. FCCU_RF_S*n* field descriptions

Field	Description
0:31 RFS[31:0]	Recoverable fault status 0 No recoverable fault latched 1 Recoverable fault latched The status bits related to the recoverable fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed.

Table 1494: FCCU RF status register channels shows FCCU **RF** status register channels.

Table 1494. FCCU RF status register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x080	FCCU_RF_S0	RFS[31:0]
0x084	FCCU_RF_S1	RFS[63:32]
0x088	FCCU_RF_S2	RFS[95:64]
0x08C	FCCU_RF_S3	RFS[127:96]

75.6.7 FCCU RF Key register (FCCU_RFK)

The FCCU_RFK register implements the key access to clear the status flags of the FCCU_RF_S*n* register.

The status bits of the FCCU_RF_S*n* register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_RFK register
2. Clear the status (flag) bit RFSx

The key must be written for each FCCU_RF_S*n* clear operation.

Note: There should not be any other operation in between the above steps.

The FCCU_RFK register is not readable; a 0x0000_0000 value is always returned in case of read operation.

Offset: 0x090 Access: User Write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	RFK_VAL															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	RFK_VAL															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1538. FCCU RF Key register (FCCU_RFK)

Table 1495. FCCU_RFK field descriptions

Field	Description
0:31 RFK_VAL	recoverable fault key = 0xAB34_98FE

75.6.8 FCCU RF Enable Register *n* (FCCU_RF_E*n*)

The FCCU_RF_E*n* registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_RF_S*n* register is anyway set, only the reaction is masked.

These registers are writable only in the CONFIG state.

Offset: 0x094 + *n**0x4 (*n* = 0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	RFE31(2)	RFE30(2)	RFE29(2)	RFE28(2)	RFE27(2)	RFE26(2)	RFE25(2)	RFE24(2)	RFE23(2)	RFE22(2)	RFE21(2)	RFE20(2)	RFE19(2)	RFE18(2)	RFE17(2)	RFE16(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	RFE15(2)	RFE14(2)	RFE13(2)	RFE12(2)	RFE11(2)	RFE10(2)	RFE9(2)	RFE8(2)	RFE7(2)	RFE6(2)	RFE5(2)	RFE4(2)	RFE3(2)	RFE2(2)	RFE1(2)	RFE0(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 1497* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 1539. FCCU RF Enable Register *n* (FCCU_RF_E*n*)

Table 1496. FCCU_RF_E*n* field descriptions

Field	Description
0:31 RFE[31:0]	Recoverable fault enable 0 No actions following the respective recoverable fault assertion 1 FCCU moves to ALARM or FAULT state

Table 1497: FCCU RF_En enable register channels shows FCCU **RF_E*n*** enable register channels.

Table 1497. FCCU RF_E*n* enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x094	FCCU_RF_E0	RFE[31:0]
0x098	FCCU_RF_E1	RFE[63:32]
0x09C	FCCU_RF_E2	RFE[95:64]
0x0A0	FCCU_RF_E3	RFE[127:96]

75.6.9 FCCU RF Timeout Enable Register *n* (FCCU_RF_TOE*n*)

The FCCU_RF_TOE*n* registers enable a transition from the NORMAL state into the ALARM state if the respective recoverable fault is enabled (RFEx and RFTOEx are set). In case the respective timeout is disabled (RFTOEx is cleared) and the recoverable fault is enabled (RFEx is set) the FCCU moves into the FAULT state if the related recoverable fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

These registers are writable only in the CONFIG state.

Offset: 0x0A4 + *n**0x4 (*n* = 0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	RFTOE31(2)	RFTOE30(2)	RFTOE29(2)	RFTOE28(2)	RFTOE27(2)	RFTOE26(2)	RFTOE25(2)	RFTOE24(2)	RFTOE23(2)	RFTOE22(2)	RFTOE21(2)	RFTOE20(2)	RFTOE19(2)	RFTOE18(2)	RFTOE17(2)	RFTOE16(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	RFTOE15(2)	RFTOE14(2)	RFTOE13(2)	RFTOE12(2)	RFTOE11(2)	RFTOE10(2)	RFTOE9(2)	RFTOE8(2)	RFTOE7(2)	RFTOE6(2)	RFTOE5(2)	RFTOE4(2)	RFTOE3(2)	RFTOE2(2)	RFTOE1(2)	RFTOE0(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 1499* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 1540. FCCU RF Timeout Enable Register *n* (FCCU_RF_TOE*n*)

Table 1498. FCCU_RF_TOE*n* field descriptions

Field	Description
0:31 RFTOE[31:0]	Fault timeout enable 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled.

Table 1499: FCCU RF timeout enable register channels shows FCCU **RF** timeout enable register channels.

Table 1499. FCCU RF timeout enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x0A4	FCCU_RF_TOE0	RFTOE[31:0]
0x0A8	FCCU_RF_TOE1	RFTOE[63:32]
0x0AC	FCCU_RF_TOE2	RFTOE[95:64]
0x0B0	FCCU_RF_TOE3	RFTOE[127:96]

75.6.10 FCCU RF Timeout Register (FCCU_RF_TO)

The FCCU_RF_TO register defines the preset value of the timer for the recovery of the recoverable faults (enabled). Once FCCU enters in ALARM state, following the assertion of a recoverable fault enabled (RFEx and RFTOEx are set), the timer starts the count down.

If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state. The alarm timeouts value should be programmed less than the FOSU_COUNT, else destructive resets may be generated by FOSU timeout.

This register is writable only in the CONFIG state.

Offset: 0x0B4 Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TO															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO															
Reset:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. Writable only in the CONFIG state.

Figure 1541. FCCU RF Timeout Register (FCCU_RF_TO)

Table 1500. FCCU_RF_TO field descriptions

Field	Description
0:31 TO	Recoverable fault timeout Timeout = (TO) x $T_{RC16MHz}$

75.6.11 FCCU CFG Timeout Register (FCCU_CFG_TO)

The FCCU_CFG_TO register defines the preset value of the watchdog timer for the recovery from the CONFIG state. Once FCCU enters in CONFIG state, following a SW request (OP1 opcode), the watchdog timer is initialized and starts the countdown if the reset is not asserted.

If the configuration is not completed within the timeout, the FCCU moves automatically from the CONFIG state to the NORMAL state and the default values for all the configuration register is restored. Configuration registers are those registers which can be written only in CONFIG state. Refer to *Table 1485: FCCU memory map*. The watchdog timeout is clocked with the RC oscillator clock (16 MHz). The default timeout value is 4.096 ms. Longer activation of CONFIG state can lead to resets if a failure is indicated during the time the FCCU is in CONFIG state due to FOSU.

The FCCU_CFG_TO register is writable in any state excluded the CONFIG state as follows the execution of the OP1 opcode (NORMAL to CONFIG state) and until the completion of the OP2 opcode (CONFIG to NORMAL state).

In case of watchdog timeout the FCCU_CFG_TO register is not accessible until the OP14 operation (CONFIG to NORMAL) has been completed.

Offset: 0x0B8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TO		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 1542. FCCU CFG Timeout Register (FCCU_CFG_TO)

Table 1501: FCCU_CFG_TO field descriptions shows FCCU_CFG_TO field descriptions.

Table 1501. FCCU_CFG_TO field descriptions

Field	Description
29:31 TO	Configuration timeout Timeout = $T_{RC16MHz}$ x $2^{(TO + 10)}$ 000 Timeout = 64 µs ... 111 Timeout = 8.192 ms

75.6.12 FCCU IO Control Register (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the FCCU Error output pin[1] output level when the FCCU is configured in “Test1” or “Test0” fault output mode (FCCU_CFG.FOM)
- to control the FCCU Error output pin[0] output level when the FCCU is configured in “Test1” or “Test2” fault output mode (FCCU_CFG.FOM)
- to observe the FCCU Error inputs level

Table 1502: Bi-stable encoding shows Bi-stable encoding.

Table 1502. Bi-stable encoding

Mode = FCCU_CFG.FOM	FCCU_EOUT[0]	FCCU_EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

Note: Due to the resynchronization stage of the EOUT interface, there is a latency of a few IRCOSC clock cycles following a write/read operation of the FCCU_EINOUT register.

Offset: 0x0BC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	EIN1(1)	EIN0(1)	0	0	EOUT1	EOUT0
W															EOUT1	EOUT0
Reset:	0	0	0	0	0	0	0	0	0	0	0/1(2)	0/1(2)	0	0	0	0

1. FCCU Error input x depends on the PAD configuration, that is, internal pull-up enabled as default or pull-down and therefore EINx bit is 1 or 0 upon reset.
2. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 1543. FCCU IO Control Register (FCCU_EINOUT)

Table 1503. FCCU_EINOUT field descriptions

Field	Description
26 EIN1	Error input 1 0 when FCCU Error input[1] = 0 1 when FCCU Error input[1] = 1 The EIN1 captures the respective FCCU Error input[1] signal.
27 EIN0	Error input 0 0 when FCCU Error input[0] = 0 1 when FCCU Error input[0] = 1 The EIN0 captures the FCCU Error input[0] signal.
30 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => FCCU Error output[1] configured in output mode). 0 force FCCU Error output[1] = 0 1 force FCCU Error output[1] = 1 The EOUT1 set/clear the respective FCCU Error output[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a “do not care” value.
31 EOUT0	Error out 0 (significant only if the FCCU_CFG.FOM = Test1 or Test2 => FCCU Error output[0] configured in output mode) 0 force FCCU Error output[0] = 0 1 force FCCU Error output[0] = 1 The EOUT0 set/clear the respective FCCU Error output[0] signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a “do not care” value.

75.6.13 FCCU Status Register (FCCU_STAT)

The FCCU_STAT register includes the FCCU status. The FCCU finite state machine (FSM) operates by the RC oscillator clock asynchronous with the System clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Offset: 0x0C0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PhysicError Pin		ESTAT	STATUS		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1544. FCCU Status Register (FCCU_STAT)

Table 1504. FCCU_STAT field descriptions

Field	Description
26:27 PhysicErrorPin	Physical Error Pin state PhysicErrorPin[26] corresponds to error pin 0, PhysicErrorPin[27] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0 Error Pin is at logical 0 1 Error Pin is at logical 1 During Fault state, a static or toggling value is observed based on selected protocol. **Note:** This field indicates the real time status of the Error pin as driven by FCCU (at FCCU boundary) to the pad and not what is seen at/from the pad, which is based on different IPs (such as SIUL2) configurations of the SoC.
28 ESTAT	Current Error Pin Status SW can write this bit with a different value. The new value is valid for 1 clock cycle, thereafter it returns to indicate the actual value being driven from FCCU. For example in nonfaulty states, this bit reads 0 and can be set for one cycle to 1. Similarly in faulty state, this bit reads 1 and sw can write it to 0 and after one cycle the bit reads 1 again. 0 System is OPERATIONAL (OK). 1 System is in FAULT state.This state is taken from the FCCU Eout logic (That is, as late as possible).
29:31 STATUS	FCCU Status 000 NORMAL state 001 CONFIG state 010 ALARM state 011 FAULT state Other: UNKNOWN state

75.6.14 FCCU NA Freeze Status Register (N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the recoverable source (FCCU recoverable fault x) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0C4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	NAFS							
W									w1c							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 1545. FCCU NA Freeze Status Register (N2AF_STATUS)

Table 1505. N2AF_STATUS field descriptions

Field	Description
24:31 NAFS	Normal to Alarm Frozen Status 0x00 No transition from NORMAL to ALARM state 0x01 NORMAL to ALARM state transition cause => FCCU recoverable fault[0] 0x02 NORMAL to ALARM state transition cause => FCCU recoverable fault[1] 0x03 NORMAL to ALARM state transition cause => FCCU recoverable fault[2] ... 0x80 NORMAL to ALARM state transition cause => FCCU recoverable fault[127] 0xFF NORMAL to ALARM state transition cause => multiple FCCU recoverable faults

75.6.15 FCCU AF Freeze Status Register (A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (recoverable fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0C8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	AF_SRC		AFFS							
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 1546. FCCU AF Freeze Status Register (A2FF_STATUS)

Table 1506. A2FF_STATUS field descriptions

Field	Description
22:23 AF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple or recoverable, or both, faults
24:31 AFFS	Alarm to Fault Frozen Status 0x00 No transition from ALARM to FAULT state 0x01 ALARM to FAULT state transition cause => FCCU recoverable fault[0] timeout 0x02 ALARM to FAULT state transition cause => FCCU recoverable fault[1] timeout 0x03 ALARM to FAULT state transition cause => FCCU recoverable fault[2] timeout ... 0x80 ALARM to FAULT state transition cause => FCCU recoverable fault[127] timeout 0xFF ALARM to FAULT state transition cause => multiple FCCU recoverable faults timeout

75.6.16 FCCU NF Freeze Status Register (N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the recoverable fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0CC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	NF_SRC		NFFS							
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 1547. FCCU NF Freeze Status Register (N2FF_STATUS)

Table 1507. N2FF_STATUS field descriptions

Field	Description
22:23 NF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple and recoverable faults
24:31 NFFS	Normal to Fault Frozen Status 0x00 No transition from NORMAL to FAULT state 0x01 NORMAL to FAULT state transition cause => FCCU recoverable fault[0] 0x02 NORMAL to FAULT state transition cause => FCCU recoverable fault[1] 0x03 NORMAL to FAULT state transition cause => FCCU recoverable fault[2] ... 0x80 NORMAL to FAULT state transition cause => FCCU recoverable fault[127] 0xFF NORMAL to FAULT state transition cause => multiple FCCU recoverable faults

75.6.17 FCCU FA Freeze Status Register (F2A_STATUS)

The F2A_STATUS register contains a unique code to identify the recoverable fault source (recoverable fault x) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the F2A_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the N2FF_STATUS register

The SW application executes the F2A_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0D0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	FAFS							
W									w1c							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 1548. FCCU FA Freeze Status Register (F2A_STATUS)

Table 1508. F2A_STATUS field descriptions

Field	Description
24:31 FAFS	Fault to Normal Frozen Status 0x00 No transition from FAULT to NORMAL state 0x01 FAULT to ALARM state transition cause => FCCU recoverable fault[0] 0x02 FAULT to ALARM state transition cause => FCCU recoverable fault[1] 0x03 FAULT to ALARM state transition cause => FCCU recoverable fault[2] ... 0x80 FAULT to ALARM state transition cause => FCCU recoverable fault[127] 0xFF FAULT to ALARM state transition cause => multiple FCCU recoverable faults

75.6.18 FCCU RF Fake Register (FCCU_RFF)

The FCCU_RFF register contains a unique code to set a recoverable fault in mutually exclusive mode by the external FAULT interface signal setting. It allows the SW emulation of the recoverable faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The fault injection mechanism is optional and requires a proper management as described in *Section 75.5.7: FAULT interface*. The reaction following a fake recoverable fault cannot be masked. The FCCU_RFF is a write-only register with a set of codes corresponding to each recoverable fault injection.

Offset: 0x0DC Access: User Write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0							
W										FRFC						
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1549. FCCU RF Fake Register (FCCU_RFF)

Table 1509. FCCU_RFF field descriptions

Field	Description
25:31 FRFC	Fake recoverable fault code 0x00 Fake recoverable fault injection at recoverable fault source 0 0x01 Fake recoverable fault injectionat recoverable fault source 1 0x02 Fake recoverable fault injection at recoverable fault source 2 … 0x7F Fake recoverable fault injection at recoverable fault source 127 **Note:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results.

75.6.19 FCCU IRQ Status Register (FCCU_IRQ_STAT)

The FCCU_IRQ_STAT register defines the FCCU interrupt status register related to the following events:

- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The external interrupt line (Interrupt request) is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Offset: 0x0E0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	Reserved	Reserved	NMI_STAT	ALRM_STAT	CFG_TO_STAT
W																w1c
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1550. FCCU IRQ Status Register (FCCU_IRQ_STAT)

Table 1510. FCCU_IRQ_STAT field descriptions

Field	Description
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

75.6.20 FCCU IRQ Enable Register (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The external interrupt line “Interrupt request (miscellaneous conditions)” is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

Offset: 0x0E4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CFG_TO_IEN
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1551. FCCU IRQ Enable Register (FCCU_IRQ_EN)

Table 1511. FCCU_IRQ_EN field descriptions

Field	Description
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled. 1 Configuration timeout interrupt enabled.

75.6.21 FCCU XTMR Register (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer or EOUT pin low counter. These timers are clocked on the IRCOSC clock.

EOUTMR is the EOUT pin in logic 0 counter. After EOUT has been set to low (by SW or by a real fault), it loads (a value corresponding to T_min = 250 µs + value programmed in FCCU_DELTA_T register) and then starts down-counting and becomes 0 on the expiry of T_min window. It reloads on occurrence of another fault event which has EOUT signaling enabled. T_min feature is available only in Bi-Stable

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
   a) OP17
   b) OP19
   c) OP20
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU_XTMR register

Table 1513: Timer state/value shows Timer state/value.

Offset: 0x0E8 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	XTMR_VAL															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR_VAL															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1552. FCCU XTMR Register (FCCU_XTMR)

Table 1512. FCCU_XTMR field descriptions

Field	Description
0:31 XTMR_VAL	Alarm/Watchdog/Safe request timer The current timer value is measured in IRCOSC clock cycles. These bits can be read by the software.

Table 1513. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	0x0000_0000	Initial value	Running	Idle/End of count
CFG	Running	0x0001_FFFF	0x0001_FFFF	0x0001_FFFF
EOUTMR	0x0000_0000	0x0000_0000	0x0000_0000	Running

75.6.22 FCCU MCS Register (FCCU_MCS)

The FCCU_MCS register contains a queue of the last four states of the MC. MCS0 is the latest one, while MCS3 is the oldest one. In addition a qualifier indicates if the FCCU is in the FAULT state when the MC state has been captured. The MC state is synchronous to the system clock and provided by a different module while the FCCU state is synchronous to the IRCOSC, therefore some uncertainty must be considered regarding the FAULT state indication.

Offset: 0x0EC Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0	0	MCS3				VL2	FS2	0	0	MCS2			
W																
Reset:	0/1(1)	0/1(1)	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0	0	MCS1				VL0	FS0	0	0	MCS0			
W																
Reset:	0/1(1)	0/1(1)	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 1553. FCCU MCS Register (FCCU_MCS)

Table 1514. FCCU_MCS field descriptions

Field	Description
0 VL3	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS3, FS3 fields are not significant. 1 MCS3, FS3 fields are significant.
1 FS3	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS3 field captured in any state different from the FAULT state. 1 MCS3 field captured in FAULT state.
4:7 MCS3	State of MC modules The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, refer to the description of the MC modules in the chip reference manual.
8 VL2	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS2, FS2 fields are not significant 1 MCS2, FS2 fields are significant
9 FS2	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS2 field captured in any state different from the FAULT state 1 MCS2 field captured in FAULT state

Table 1514. FCCU_MCS field descriptions (continued)

Field	Description
12:15 MCS2	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, refer to the description of the MC modules in the chip reference manual.
16 VL1	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS1, FS1 fields are not significant. 1 MCS1, FS1 fields are significant.
17 FS1	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS1 field captured in any state different from the FAULT state. 1 MCS field captured in FAULT state.
20:23 MCS1	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, refer to the description of the MC modules in the chip reference manual.
24 VL0	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS0, FS0 fields are not significant. 1 MCS0, FS0 fields are significant.
25 FS0	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS0 field captured in any state different from the FAULT state. 1 MCS0 field captured in FAULT state.
28:31 MCS0	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, refer to the description of the MC modules in the chip reference manual.

75.6.23 FCCU Transient Register (FCCU_TRANS_LOCK)

The FCCU_TRANS register is used for unlocking configuration by writing 0xBC. This register is available only in supervisor mode. This register is write only.

Offset: 0x0F0 Access: Supervisor Write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0								
W									TRANSKEY							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1554. FCCU Transient Register (FCCU_TRANS_LOCK)

Table 1515. FCCU_TRANS_LOCK field descriptions

Field	Description
24:31 TRANSKEY	Transition Unlocking Key value 0x00 No change ... 0xBB No change 0xBC Configuration unlocked 0xBD No change ... 0xFF No change

75.6.24 FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing 0xFF. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.

Offset: 0x0F4 Access: Supervisor Write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0								
W									PERMNTKEY							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1555. FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

Table 1516. FCCU_PERMNT_LOCK field descriptions

Field	Description
24:31 PERMNTKEY	Transition Locking Key value 0xFF Configuration Locked Any other value: No Change

75.6.25 FCCU Delta T Register (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds. This register can be written only in CONFIG state.

Offset: 0x0F8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	Reserved													
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	DELTA_T_VAL													
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1556. FCCU Delta T Register (FCCU_DELTA_T)

Table 1517. FCCU_DELTA_T field descriptions

Field	Description
18:31 DELTA_T_VAL	Value of Delta_T (in microseconds) Max. value can be 10,000 µs (10 ms)

75.6.26 FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm. This register is writable only in CONFIG state.

Offset: 0x0FC + *n**0x4 (*n* = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	IRQEN31[1]	IRQEN30[1]	IRQEN29[1]	IRQEN28[1]	IRQEN27[1]	IRQEN26[1]	IRQEN25[1]	IRQEN24[1]	IRQEN23[1]	IRQEN22[1]	IRQEN21[1]	IRQEN20[1]	IRQEN19[1]	IRQEN18[1]	IRQEN17[1]	IRQEN16[1]
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	IRQEN15[1]	IRQEN14[1]	IRQEN13[1]	IRQEN12[1]	IRQEN11[1]	IRQEN10[1]	IRQEN9[1]	IRQEN8[1]	IRQEN7[1]	IRQEN6[1]	IRQEN5[1]	IRQEN4[1]	IRQEN3[1]	IRQEN2[1]	IRQEN1[1]	IRQEN0[1]
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 1519* for register offset to channel number relationship.

Figure 1557. FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

Table 1518. FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0:31 IRQEN[31:0]	IRQ alarm enable 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

Table 1519: FCCU IRQ alarm enable register channels shows FCCU IRQ alarm enable register channels.

Table 1519. FCCU IRQ alarm enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x0FC	FCCU_IRQ_ALARM_EN0	IRQEN[31:0]
0x100	FCCU_IRQ_ALARM_EN1	IRQEN[63:32]
0x104	FCCU_IRQ_ALARM_EN2	IRQEN[95:64]
0x108	FCCU_IRQ_ALARM_EN3	IRQEN[127:96]

75.6.27 FCCU NMI Enable registers *n* (FCCU_NMI_EN*n*)

These registers enable the NMI. These registers are writable only in CONFIG state.

Offset: 0x10C + *n**0x4 (*n* = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	NMIEN31(1)	NMIEN30(1)	NMIEN29(1)	NMIEN28(1)	NMIEN27(1)	NMIEN26(1)	NMIEN25(1)	NMIEN24(1)	NMIEN23(1)	NMIEN22(1)	NMIEN21(1)	NMIEN20(1)	NMIEN19(1)	NMIEN18(1)	NMIEN17(1)	NMIEN16(1)
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	NMIEN15(1)	NMIEN14(1)	NMIEN13(1)	NMIEN12(1)	NMIEN11(1)	NMIEN10(1)	NMIEN9(1)	NMIEN8(1)	NMIEN7(1)	NMIEN6(1)	NMIEN5(1)	NMIEN4(1)	NMIEN3(1)	NMIEN2(1)	NMIEN1(1)	NMIEN0(1)
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 1521* for register offset to channel number relationship.

Figure 1558. FCCU NMI Enable register *n* (FCCU_NMI_EN*n*)

Table 1520. FCCU_NMI_EN*n* field descriptions

Field	Description
0:31 NMIEN[31:0]	NMI enable 0 NMI is disabled for error (RF) source x. 1 NMI is enabled for error (RF) source x.

Table 1521: FCCU NMI enable register channels shows FCCU NMI enable register channels.

Table 1521. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x10C	FCCU_NMI_EN0	NMIEN[31:0]
0x110	FCCU_NMI_EN1	NMIEN[63:32]
0x114	FCCU_NMI_EN2	NMIEN[95:64]
0x118	FCCU_NMI_EN3	NMIEN[127:96]

75.6.28 FCCU EOUT Signaling Enable registers *n* (FCCU_EOUT_SIG_EN*n*)

These registers enable the error out signaling. Disabling of EOUT signaling is possible for Bi-stable protocol only. This register is writable only in CONFIG state.

Offset: 0x11C + n*0x4 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	EOUTEN31(1)	EOUTEN30(1)	EOUTEN29(1)	EOUTEN28(1)	EOUTEN27(1)	EOUTEN26(1)	EOUTEN25(1)	EOUTEN24(1)	EOUTEN23(1)	EOUTEN22(1)	EOUTEN21(1)	EOUTEN20(1)	EOUTEN19(1)	EOUTEN18(1)	EOUTEN17(1)	EOUTEN16(1)
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	EOUTEN15(1)	EOUTEN14(1)	EOUTEN13(1)	EOUTEN12(1)	EOUTEN11(1)	EOUTEN10(1)	EOUTEN9(1)	EOUTEN8(1)	EOUTEN7(1)	EOUTEN6(1)	EOUTEN5(1)	EOUTEN4(1)	EOUTEN3(1)	EOUTEN2(1)	EOUTEN1(1)	EOUTEN0(1)
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 1523* for register offset to channel number relationship.

Figure 1559. FCCU EOUT Signaling Enable register *n* (FCCU_EOUT_SIG_EN*n*)

Table 1522. FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0:31 EOUTEN[31:0]	EOUT signaling enable 0 EOUT signaling is disabled for error (RF) source x for bistable protocol. Error pins are as if in nonfaulty state.(1) 1 EOUT signaling is enabled for error (RF) source x.

1. In case of time switching and dual rail protocols, the error out signaling remains active even if the EOUT flag is 0b.

Table 1523: FCCU EOUT signaling enable register channels shows FCCU EOUT signaling enable register channels.

Table 1523. FCCU EOUT signaling enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x11C	FCCU_EOUT_SIG_EN0	EOUTEN[31:0]
0x120	FCCU_EOUT_SIG_EN1	EOUTEN[63:32]
0x124	FCCU_EOUT_SIG_EN2	EOUTEN[95:64]
0x128	FCCU_EOUT_SIG_EN3	EOUTEN[127:96]

75.7 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary error notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after an error arrives.

The intention of the FOSU is to provide a secondary error reaction path in most cases the FCCU fails but not to needlessly propagate an error which is already handled by the FCCU into a full chip reset. Only a failed primary error reaction (that is FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction. It is a "static" input in the sense that it does not change after FCCU configuration. This signal is automatically de-asserted whenever the FCCU moves to CONFIG state and hence FOSU shall not be able to monitor faults occurring in CONFIG state.The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted (that is an enabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state. Also, the case of a continuously incoming disabled fault being enabled later, is not monitored.

The FOSU contains a timer with a duration of FOSU_COUNT,driven by the IRCOSC.The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FCCU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: Long or short functional reset.
- IRQ: NMI or Alarm.
- Error out triggered (by FCCU or by SW).

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still ‘stuck’ in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static “fccu program to do nothing” signal is de-asserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.
The FOSU's failure indicator output is connected to one of the MC_RGM's ‘destructive’ reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as it is used by the FCCU (System reset), which is asserted on power-on, ‘destructive’, and long external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is de-asserted.

The following conditions show a use case of the FOSU:

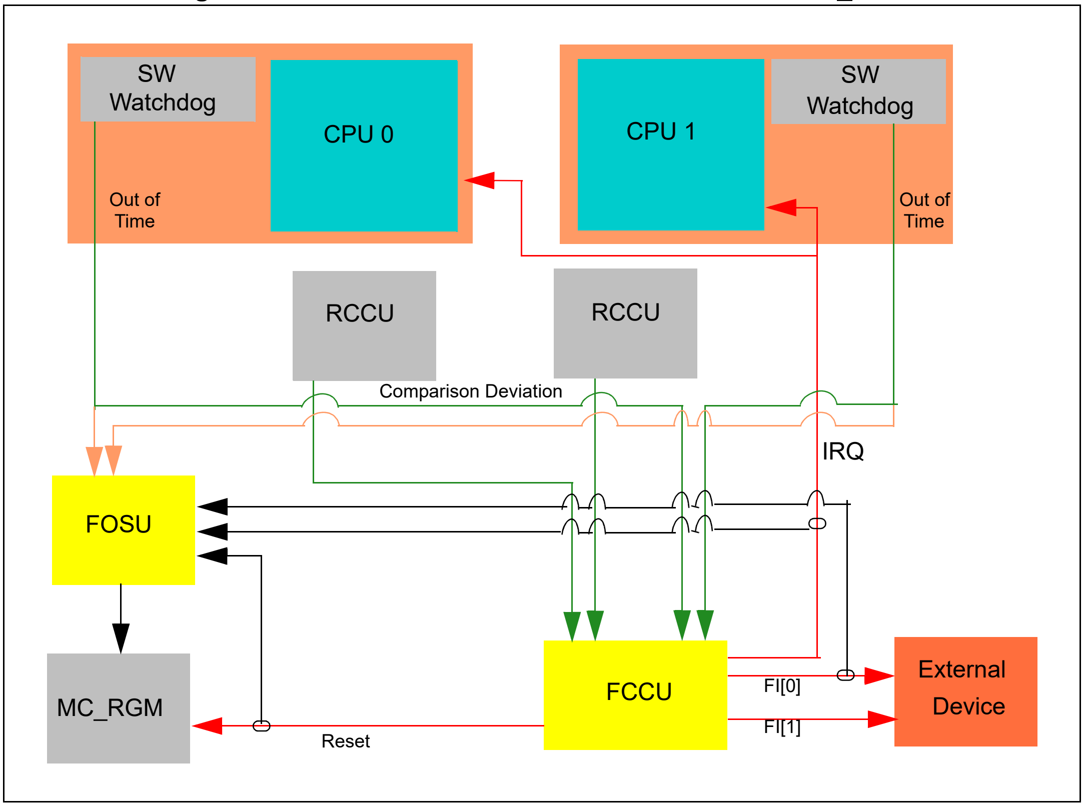
1. The user configures the channel x of the FCCU to react with a functional reset, either long or short.
2. The user sets the EOUTEN[x] flag of EOUT_SIG_EN register to 0.
3. The software does not clear the FCCU status flag related to the fault x after its first occurrence.
4. The fault x triggers again after the functional reset. Under these circumstances, the FOSU triggers a destructive reset after the FOSU timeout.

The rationale behind is that the functional reset is not able to clear the source of the fault. If the software can't react to the second occurrence of the fault before the FOSU timeout expires, the FOSU triggers a destructive reset.

If the users want to increase the availability of the system by avoiding the destructive reset, they can set the EOUTEN[x] flag of EOUT_SIG_EN register to 1. In such a case, the FOSU does not trigger the destructive reset.

Note: FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction. All intermediate faults are masked (not monitored) during this timeout cycle.

Figure 1560. FOSU connections to the FCCU and MC_RGM



It is important to remember that there will be no FCCU reaction to an error while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

75.8 Use cases and limitations

75.8.1 Mis-configurations

The following configurations are appropriate:

1. If at least one reaction to fault is enabled by default then the corresponding fault should also be enabled by default.
2. Enable a fault but disabling all reactions is not a meaningful configuration from safety point of view.
3. Disable the fault and also its reactions. Applicable when a specific fault line is not of interest in a specific application scenario.

Example: FCCU goes into ALARM state due to a failure, but the IRQ is disabled for that failure.

4. Software reset is optional.
5. The user must program the FCCU for a higher timeout value than the value hard-coded for FOSU. In this way FCCU reacts later than FOSU expects and as a result, FOSU generates a destructive reset request on its timeout, if a fault occurred when the FCCU was in CONFIG state.

75.8.2 Recommendations to configure FCCU

1. After Power on reset, where both system and FCCU get reset, the following steps could be followed to configure FCCU:
   a) Check and clear any pending fault status.
   b) Verify FCCU is in NORMAL state, else repeat step (a) above.
   c) Configure FCCU.
2. After functional reset of the system, arising out of reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a) If active, wait for the Error out Tmin to expire.
   b) Check and clear fault status.
   c) Error pin moves to “non faulty” state, once fault status is cleared and Tmin expires.
   d) Reset FCCU by register bit in RGM. All FCCU registers and Error pin initialize to default state.
   e) Read and verify default value in FCCU_RF_Ex to ascertain reset of FCCU.
   f) Check and clear any new pending fault status until FCCU is in NORMAL state.
   g) Reconfigure FCCU.

Note: Software reset of FCCU by RGM_PRST5.FCCU_RST bit is optional since FCCU registers are not initialized during functional reset. RGM_PRST5.FCCU_RST has the same effect on FCCU as power on reset. If the bit is activated, it can change the state of the error pin, if the current state of the pin is different from default.

76 Self-Test Control Unit (STCU2)

76.1 Introduction

The self-test control unit (STCU2) is a comprehensive programmable hardware module that controls the self-test sequence applied both during the offline or online, or both, conditions. The hardware can manage the Memory (SRAM or ROM) built-in self-test (MBIST) blocks of the device. The STCU2 includes the SSCM DCF bus to load the self-test parameters (MBIST scheduling activity, unrecoverable faults (UF) or recoverable faults (RF) management, CRC and MISR expected values, PLL management, and so on) from Flash memory during the offline self-test phase. This interface is only able to write the configuration parameters and start the self-test execution once the STCU2 global reset has been applied. Register access by software is granted by an IPS interface to check the results of the offline self-test and to load or check the execution of the online self-test.

The STCU2 includes:

- A CRC block that can be optionally enabled to monitor the internal critical signals during the self-test run to increase the STCU2’s self-checking capability.
- A programmable watchdog timer (WDG) to check the MBIST self-test operations have completed within the assigned time slot or the STCU2 RUN or BYPASS bits have been programmed before watchdog timeout.
- Direct PLL control during the offline self-test sequence.
- Two interrupt lines enabled only during the online self-test to simplify the software handling of self-test execution end.
- A set of registers to collect the MBIST results during offline and online self-test.

To increase the security of this module, a different security key sequence is also required during offline and online self-test to unlock write access to the STCU2 registers. The STCU2 also implements two switch off power-saving mechanisms:

- The global STCU2 clock after self-test sequence is completed.
- The register ITF clock when the WDG timeout is detected after a double security key sequence is applied during the application run (available only when IPS write access is enabled otherwise the clock is off).

76.2 Main features

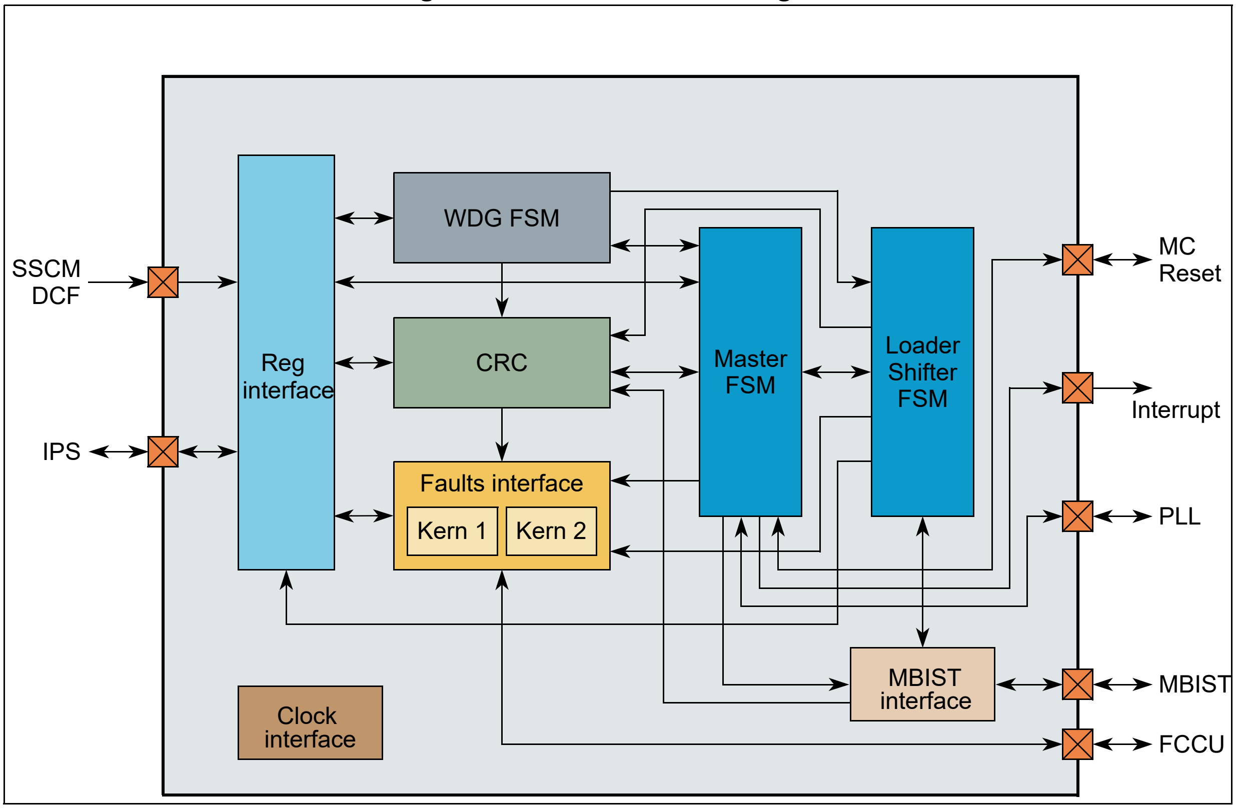
- Double registers interface (SSCM DCF with priority over IPS).
- SSCM DCF write once register interface.
- IPS read and write (depending on SSCM enable parameters and STCU_CFG[WRP] bit) register interface.
- Programmable scheduler for MBIST execution.
- MBIST concurrent/sequential execution.
- 32-bit CRC for STCU2 self-checking capability (enabled by STCU_CFG[CRCEN] bit).
- Online hardware self-test sequence abort capability.
- Programmable internal clock prescaler to reduce internal and MBIST TCK clocks.
- Programmable PLL direct control during offline self-test sequence.

- PLL lock signal monitoring during offline and online self-test.
- MBIST dedicated online interrupt source generation.
- Programmable WDG timer for self-check MBIST time execution.
- Hard-coded WDG timeout to flag offline initialization errors.
- Hard-coded WDG timeout to auto-lock STCU2 write access after double key sequence.
- SRAM or ROM MBIST offline status flags.
- STCU2 internal errors offline status flag.
- SRAM or ROM MBIST online status flags.
- STCU2 internal errors online status flag.
- Online hardware abort status flag.
- SRAM or ROM MBIST offline or online programmable failure severity (UF/RF).
- STCU2 internal errors offline or online programmable failure severity (UF/RF).
- STCU2 errors (UF/RF) lines for FCCU signaling.
- Redundant UF/RF generation logic to improve safety.
- FCCU UF/RF faults injection mechanism.
- CRCE register access to perform redundant check by software.
- Self-test bypass capability.
- Protection mechanism of the global write register based on double security key.
- STCU2 global auto power-saving when:
  - Self-test is completed.
  - bypass mode is selected.
  - WDG timeout is detected.
- STCU2 ITF/WDG clock wake-up mechanism when software application writes double security key sequence (available when STCU_CFG[WRP] bit is active).
- STCU2 ITF/WDG auto power-saving when hard-coded WDG timeout is detected after software application writes double security key sequence (available when STCU_CFG[WRP] bit is active).

76.3 Block diagram

A top level diagram of the STCU2 module is given in *Figure 1561*.

Figure 1561. STCU2 block diagram



The STCU2 module includes the following submodules:

- **Reg interface**: This module includes the registers, the security key logic, the IPS and SSCM DCF bus interface.
- **CRC**: This module is the core of the self-checking architecture of the STCU2. It samples a set of selected internal signals when STCU2 is running.
- **Fault interface:** This module collects the error conditions respective to MBIST execution and STCU2 internal failures and, depending on the UF/RF configuration of each of them, sets the global STCU2 UF or RF flag. It also manages the UF/RF fault lines and the set or clear injection mechanism provided by the FCCU. To improve the intrinsic safety of this critical logic, the generation logic is duplicated.
- **Clock Block:** This module manages the internal and the MBIST TCK clock prescaler, the internal clock gating power saving and the wake-up clock feature.
- **WDG FSM:** This module includes two different watchdogs:
  – A hard-coded Watchdog used to auto-lock the STCU2 access, forcing a reset condition on the double security key registers.
  – A dual function watchdog:

    - After a reset event initializes the STCU2, it is used as a hard-coded watchdog timeout.

    - During MBIST runs, it is used as a programmable watchdog timer to check the MBIST has completed in the designated time slot.
- **STCU2 Master FSM**: This module includes the main FSM of the STCU2 used to coordinate and schedule all the operations performed during the self-test sequence.

- **STCU2 Load Shifter FSM:** This module includes the common shifter register and the respective state machine used to program the MBIST registers and read back the data to be checked at the end of each self-test operation.
- **MBIST interface**: This module includes the interface between the MBIST controller and STCU2 controllers.

76.3.1 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers.
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers.
- Byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers.
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode under the following circumstances:

- The self-test sequence parameters are loaded before offline self-test is executed.
- The IPS access on some specific registers (see *Section 76.5: Register description*) is allowed by software only when the bit STCU_CFG[WRP] is cleared.
- Write operations are not supported and all STCU2 registers can only be read in all other circumstances except for the following R/W bits: STCU_CFG[WRP], STCU_ERR_STAT[UFSF] and STCU_ERR_STAT[RFSF]. See the respective register descriptions for additional details.

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral.
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register.
- Any write operation on double security key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other accesses to STCU2 registers).
- Any write operation performed on a register when the double security keys have not been applied.
- Any write operation performed on registers when the STCU_CFG[WRP] bit is set and the access is performed through software (IPS interface).
- Any write operation performed through software (IPS interface) on row registers.
- Any write operation performed offline on registers where writes are only allowed during the online self-test phase when STCU_CFG[WRP] = 1.
- Any write operation performed on read only registers.

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

A transfer error is not generated for write operations to bits marked *Reserved*.

For more details regarding register protection features, see *Section 76.5: Register description*.

76.4 Functional description

76.4.1 FSM description

The module has three state machines that work together:

- Master State Machine: the core unit of the STCU2 module; it coordinates all the self-test operations and the other state machines.
- Loader/Shifter State Machine: used to program the MBIST parameters and to retrieve the respective results depending on the parameters stored in the STCU2 registers and under the control of the master state machine.
- Watchdog State Machine: evaluates all the schedule time for MBIST and the timeouts in case of incorrect STCU2 programming.

76.4.2 Reset management

The asynchronous reset line initializes the STCU2 module status, forcing the offline self-test condition and the cleaning up of all the registers and state machines of the module.

The reset signals generated by the STCU2 module at the end of the self-test execution phase depends on the self-test conditions reported in the following subsections.

In case of unrecoverable fault during offline self-test execution, the STCU2 generates a request of destructive reset to the associated MC_RGM module.

76.4.2.1 Offline reset generation

At the end of the offline self-test execution, the STCU2 raises the global functional reset. In case of unrecoverable fault during offline self-test execution, the STCU2 generates a request of destructive reset to the MC_RGM module.

76.4.2.2 Online reset generation (only MBIST)

The online execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is because the software application has to manage all the possible consequences respective to the write operations performed by the MBIST execution other than prevent any read/write operation during the self-test execution. If necessary, the software must also restore the memory content (only for RAM) at the end of the self-test execution.

76.4.3 MBIST scheduling

The STCU2 module is designed to be very flexible, allowing the programming of the parallel or serial execution of the MBIST depending on the power, timing and coverage constraints. The limitation in the programming flexibility is described in *Section 76.6.4: Design implementation information*.

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU_CFG[PTR]. The first MBIST is mapped at the address 0x10, the second on 0x11 and so on. The additional pointers are in the bitfield STCU_MB_CTRL*y*[PTR] for MBIST (where *y* is the selected MBIST), and have to be filled depending on the selected sequence of run. The additional bitfields STCU_MB_CTRL[CSM] provide the flexibility to run the chosen set of the MBIST concurrently or sequentially, or to close the linked list by setting the NIL pointer. For more details, see *Section 76.5.22: STCU2 MBIST Control k Register (STCU_MB_CTRLk)*.

76.4.4 ABORT management

The STCU2 module provides online self-test execution abort. When the abort is detected, the online self-test operation stops running and the status of the currently running MBIST is saved in the respective registers to provide intermediate results that might be useful in case of debug or a concurrent run.

76.4.4.1 Hardware ABORT management

The STCU2 enters this abort condition as a consequence of a functional reset that requires the self-test execution to stop.

When hardware abort is detected, the STCU_ERR_STAT[ABORTHW] bit is set. This flag allows the software to diagnose what occurred during the online self-test execution run.

76.4.5 PLL interface

The STCU2 module is able to directly control the PLL during the offline self-test operations depending on the status of the STCU_RUN[MBPLLEN] and STCU_RUN[LBPLLEN] as described in *Section 76.5.2: STCU2 Run Software Register (STCU_RUNSW)*.

When STCU_RUN[RUN] is set to start the offline self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configuration parameters according to the values programmed in the STCU_PLL_CFG register.

When the STCU_RUN[MBPLLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock. As soon as the lock occurs, the STCU forces the clock to switch to the PLL output clock, allowing the MBIST to proceed at the PLL output frequency. At the end of the MBIST run, the clock source reverts to the original one.

76.4.6 Interrupt interface

There is one interrupt source:

- triggered by MBIST (enabled by programming STCU_RUNSW[MBIE] = 1.

As described in *Section 76.5.2: STCU2 Run Software Register (STCU_RUNSW)*, the interrupts are generated at the end of any concurrent run. This means that in case of multiple sequential runs in the same session, the number of interrupts generated is equivalent to the number of sequential sessions, thus providing fine granularity with respect to the completed run.

The MBIFLG bit in the *STCU2 Interrupt Flag Register (STCU_INT_FLG)* registers flag interrupt condition and allows the clearing of interrupt requests. Interrupt bit to be cleared by writing the respective bit to 1 to enable the next interrupt request.

76.4.7 FCCU interface

The FCCU interface is the hardware flag mechanism to the system, indicating the occurrence of an unrecoverable failure (UF) or a recoverable failure (RF) during the self-test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. The FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of

FCCU fault injection mechanism to understand how the UF/RF set and clear mechanism works.

76.4.8 Watchdogs

The STCU2 implements three different watchdogs to ensure that operations are finished on time.

76.4.8.1 Initialization sequence

The STCU2 has two initialization operating modes:

- Program and run the self-test sequence (safety mode).
- Bypass the self-test sequence by setting the STCU_RUN[BYP] bit. If Bypass bit is configured/set by DCF it cannot be anymore changed on the device even though STCU_RUN register itself is not Write once.

If there are faults during the initialization phase which prevent the selection of one of these two operating modes, a hard-coded watchdog timeout flags the incorrect behavior. See *Section 76.5.6: STCU2 Watchdog Register Granularity (STCU_WDG)* for more details.

76.4.8.2 MBIST scheduling

The MBIST execution time is configured as described in *Chapter 76.5.6: STCU2 Watchdog Register Granularity (STCU_WDG)* to account for the overall execution time of the self-test sequence. If the selected MBISTs are not yet completed:

- During an offline self-test assigned time, the current MBISTs execution is interrupted and a failure is flagged in STCU_ERR_STAT[WDTO] and STCU_MBE1/2 registers.
- During an online self-test, the current MBISTs execution is interrupted and a failure is flagged in STCU_ERR_STAT[WDTOSW] and STCU_MBE1SW/2SW.

In case of multiple sequential runs in the same online or offline session and the timeout occurs in the middle of a sequential run, the next sequential run is skipped and the execution ends with the current updated status of the registers reported above.

In all cases, the STCU2 reset generation is managed in one of the modes described in *Section 76.4.2: Reset management*.

76.4.8.3 Write access timeout

The access on the STCU2 registers during the self-test configuration phase (both online or offline) is protected by a key mechanism to prevent any unwanted access, as described in the STCU_SKC register description. In addition to this safety feature, there is a hardware watchdog that, after 1024 STCU2 clock cycles, locks access and requires reinsertion of the STCU_SKC keys.

This feature is particularly useful if STCU_CFG[WRP] = 0 during the software self-test configuration as the software application may enable write access to the STCU2 registers.

76.4.9 CRC

The CRC block is used to increase the intrinsic coverage of the STCU2 module and implements a 32-bit ethernet protocol polynomy to evaluate the signature of the most important and critical internal signals. This feature is switched off by default but it can be turned on setting the STCU_CFG[CRCEN] bit. Once this bit is set, the expected signature has to be programmed in the STCU_CRCE register. At the end of the self-test execution, the STCU2 updates the content of the STCU_CRCR register with the evaluated signature and compares this value with the expected one previously programmed in STCU_CRCE. In case of mismatches, a UF/RF condition is flagged depending on the fault mapping selected in the STCU_ERR_FM register.

Note: The expected value of the CRC depends on multiple internal critical signals and can differ among different samples.

76.4.10 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without CPU intervention after a reset trigger event initializes the STCU2. This bus interface has priority over the IPS.

76.5 Register description

The STCU2 registers are listed in Table 1524.

Note: k represents the repeated register blocks of the multiple MBISTs: k ranges from 0 up to 34.

Table 1524. STCU2 memory map

Address offset	Register Name	Location
0x0000	STCU2 Run Register (STCU_RUN)	Section 76.5.1
0x0004	STCU2 Run Software Register (STCU_RUNSW)	Section 76.5.2
0x0008	STCU2 SK Code Register (STCU_SKC)	Section 76.5.3
0x000C	STCU2 Configuration Register (STCU_CFG)	Section 76.5.4
0x0010	STCU2 PLL Configuration Register (STCU_PLL_CFG)	Section 76.5.5
0x0014	STCU2 Watchdog Register Granularity (STCU_WDG)	Section 76.5.6
0x0018	STCU2 Interrupt Flag Register (STCU_INT_FLG)	Section 76.5.7
0x001C	STCU2 CRC Expected Status Register (STCU_CRCE)	Section 76.5.8
0x0020	STCU2 CRC Read Status Register (STCU_CRCR)	Section 76.5.9
0x0024	STCU2 Error Register (STCU_ERR_STAT)	Section 76.5.10
0x0028	STCU2 Error FM Register (STCU_ERR_FM)	Section 76.5.11
0x002C–0x0043	Reserved	
0x0044	STCU2 offline MBIST 1 Status Register (STCU_MBS1)	Section 76.5.12
0x0048	STCU2 offline MBIST 2 Status Register (STCU_MBS2)	Section 76.5.13
0x0054	STCU2 offline MBIST 1 End Flag Register (STCU_MBE1)	Section 76.5.14
0x0058	STCU2 offline MBIST 2 End Flag Register (STCU_MBE2)	Section 76.5.15
0x0064	STCU2 online MBIST 1 Status Register (STCU_MBS1SW)	Section 76.5.16
0x0068	STCU2 online MBIST 2 Status Register (STCU_MBS2SW)	Section 76.5.17
0x0074	STCU2 online MBIST 1 End Flag Register (STCU_MBE1SW)	Section 76.5.18
0x0078	STCU2 online MBIST 2 End Flag Register (STCU_MBE2SW)	Section 76.5.19

Table 1524. STCU2 memory map (continued)

Address offset	Register Name	Location
0x0084	STCU2 MBIST 1 Unrecoverable FM Register (STCU_MBUFM1)	Section 76.5.20
0x0088	STCU2 MBIST 2 Unrecoverable FM Register (STCU_MBUFM2)	Section 76.5.21
0x0094–0x05FF	Reserved	
MBIST registers (*k* = 0 to 34)		
0x0600 + *k* * 0x4	STCU2 MBIST Control Register (STCU_MB_CTRL*k*)	Section 76.5.22

76.5.1 STCU2 Run Register (STCU_RUN)

The STCU_RUN register defines the RUN bit to start the offline self-test procedure if some configuration bits are required *only* during this self-test phase.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is still active.

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BYP	MBPLLEN	0	0	0	0	0	0	0	0	RUN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1562. STCU2 Run Register (STCU_RUN)

Table 1525. STCU_RUN field descriptions

Field	Description
21 BYP	Bypass mode 0 Offline self-test is active. If the STCU_RUN[RUN] bit is not set before the hard-coded WDG timeout, the WDTO error is generated in the STCU_ERR_STAT register. 1 Offline self-test is bypassed and access to STCU2 is locked until the STCU_SCK register is written according to the request access mode.

Table 1525. STCU_RUN field descriptions (continued)

Field	Description
22 MBPLLEN	Offline MBIST with PLL0 Enabled 0 Offline MBIST is executed without using the on-chip PLL0. 1 Offline MBIST is executed enabling the on-chip PLL0 control interface selecting the parameters defined in STCU_PLL_CFG register.
31 RUN	RUN The RUN bit is automatically cleared by the STCU2 when the offline self-test procedure has been completed. 0 Idle 1 Offline self-test procedure is running

76.5.2 STCU2 Run Software Register (STCU_RUNSW)

The STCU_RUNSW register defines the RUN bit to start the online self-test procedure if some configuration bits are required *only* during this self-test phase.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU_CFG[WRP] = 0.
- Online self-test phase is active. The online condition starts when the offline condition finishes. In this condition, the system is alive and the SW can program the STCU.

Offset: 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	MBIE	0	MBSWPLLEN	0	0	0	0	0	0	0	0	RUNSW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1563. STCU2 Run Software Register (STCU_RUNSW)

Table 1526. STCU_RUNSW field descriptions

Field	Description
20 MBIE	MBIST Interrupt Enable 0 Interrupt is not generated at the end of the software MBIST execution phase of the selected Memories. 1 At the end of the software MBIST execution phase of the concurrent selected memories, the STCU2 forces an interrupt request to notify MBISTs of that concurrent selected memories are concluded. If there is a sequence of concurrent runs in the same self-test run, the interrupt request is generated at the end of each concurrent run provided that the software clears the STCU_INT_FLG[MBIFG] bit in between.
22 MBSWPLLEN	Online MBIST with PLL1 Enabled 0 Online MBIST is executed without waiting for PLL1 lock. 1 Online MBIST is executed once PLL1 is locked. PLL1 configuration used is the one specified in the PLLDIG PLL1DV register. STCU2 does not take PLL1 control but monitors the PLL1 lock signal to check if PLL1 is working correctly.
31 RUNSW	RUNSW The RUNSW bit is automatically cleared by the STCU2 when the online self-test procedure has been completed. 0 Idle 1 Online self-test procedure is running

76.5.3 STCU2 SK Code Register (STCU_SKC)

The STCU_SKC register implements the security key code mechanism needed for write access to the other STCU2 registers. In order to unlock the STCU2 access after an STCU2 asynchronous reset or the end of the STCU2 run, the software (IPS bus) or the SSCM interfaces must apply the following sequence:

- Write Key1 in the STCU_SKC register.
- Write Key2 in the STCU_SKC register.

Depending on the offline or online test step, the two keys will be different. Byte write operation is not allowed as the full key has to be recognized as one unit.

Note: Note that after the self-test sequence has been completed or the bypass feature has been enabled (by setting the STCU_RUN[BYP] bit), the SSCM interface is no longer available.

In case of invalid access or sequence (Keys1 and 2 must be applied consecutively), a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked until the correct sequence is applied.

If the STCU2 register access lasts more cycles than defined in the hard-coded WDG timeout, STCU2 write access is locked and the WDG and register ITF clocks are switched off. Also in this case the correct sequence must be applied in order to re-enable write access to the STCU2 and the WDG and register ITF clocks.

To extend the STCU2 register access cycles before the hard-coded WDG timeout expires, only Key2 needs to be applied. The effect of this write operation is to re-initialize the WDG timeout counter. Key1 must not be applied or a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked until the correct sequence is applied.

The STCU_SKC register is not readable; any attempts to read this register always returns the value 0x00000000.

Offset: 0x0008 Access: User write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	SKC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	SKC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1564. STCU2 SK Code Register (STCU_SKC)

Table 1527. STCU_SKC field descriptions

Field	Description
0:31 SKC	STCU2 security key code for offline test 0xD3FEA98B = Key1 to unlock the write access the STCU2 0x2C015674 = Key2 to unlock the write access the STCU2 STCU2 security key code for online test 0x753F924E = Key1 to unlock the write access the STCU2 0x8AC06DB1 = Key2 to unlock the write access the STCU2

76.5.4 STCU2 Configuration Register (STCU_CFG)

The STCU_CFG register includes the global configuration of the STCU2 and can be updated both in the offline and online test steps.

Access to this register is described in *Figure 1565*. It also depends on the state of the WRP bit as follows:

- When WRP = 0, the register can be written during the online self-test case without restrictions
- When WRP = 1:
  - The only bit that can be written without any restriction is WRP.
  - If software operations write to other register bits, a transfer error is raised if the value of the selected byte that is written differs from the current status of the register. This function is implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

Offset: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTR(1)								0	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	WRP	0	0	CRCEN(1)	PMOSEN(1)	MBU(1)	CLK_CFG(1)		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. You can always read this field. In the offline self-test phase, you can always write to this field. In the online self-test phase, you can write to this field only when WRP = 0.

Figure 1565. STCU2 Configuration Register (STCU_CFG)

Table 1528. STCU_CFG field descriptions

Field	Description
0:7 PTR	MBIST pointer. PTR defines the logical pointer to the first MBIST to be scheduled when the self-test procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. (0x10 to 0x10 + MBIST - 1) pointer to MBIST 0xFF pointer to NIL. No BIST execution.
23 WRP	Write Protection. 0 Specific STCU2 registers can be written through IPS bus interface after offline self-test sequence has completed. 1 Specific STCU2 registers cannot be written though IPS even after offline self-test sequence has completed, thus preventing any user application write operation.
26 CRCEN	CRC Enable comparison. 0 The CRC comparison is not performed and the STCU_ERR_STAT[CRCS][SW] and STCU_ERR_STAT[UF/RF] global flags are not updated in case of mismatches between STCU2[CRCE] and STCU2[CRCR] values. 1 The CRC comparison is performed and the status is updated in the respective flags of the STCU_ERR_STAT register
27 PMOSEN	MBIST PMOS Test Enable. 0 MBIST PMOS test is not enabled 1 MBIST PMOS test is enabled

Table 1528. STCU_CFG field descriptions (continued)

Field	Description
28 MBU	MBIST MBU Test Enabled. 0 MBIST full algorithm or reduced one without PMOS test (PMOSEN set to 0) is applied. 1 MBIST simplified multi-bit upset algorithm is used to check the RAM. If a ROM is selected, the applied algorithm will be standard one without PMOS test. ***This bit overwrites the PMOSEN feature***.
29:31 CLK_CFG	Logic, memory BIST and STCU2 core clock configuration. CLK_CFG defines the ratio between the STCU2 core clock and the TCK used to program both the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are: 000 PBRIDGE_CLK 001 PBRIDGE_CLK/2 010 PBRIDGE_CLK/3 011 PBRIDGE_CLK/4 100 PBRIDGE_CLK/5 101 PBRIDGE_CLK/6 110 PBRIDGE_CLK/7 111 PBRIDGE_CLK/8 **Note:** refer to Clocking chapter for PBRIDGE_CLK clock frequency value.

76.5.5 STCU2 PLL Configuration Register (STCU_PLL_CFG)

The STCU_PLL_CFG register defines the parameters used to program the PLL only when the offline MBISTs are performed and STCU_RUN[MBPLLEN] = 1.

In the online state, these bits are not effective. It is also possible to set the PLL during active online mode to prevent any potential functionality issues.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is active.

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PLLODF						0	0	0	0	0	PLLIDF		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	PLLLDF						
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1566. STCU2 PLL Configuration Register (STCU_PLL_CFG)

Note: *f_{RC} is the frequency if the internal RC oscillator (that is 16 MHz). f_{PLLin} must be higher than 8 MHz.*

Table 1529. STCU_PLL_CFG field descriptions

Field	Description
2:7 PLLODF	PLL Output Division Factor. The value of this field drives the output division factor of the PLL embedded in the device. Refer to the PLL specifications to define the exact mapping between the PLLODF value and the respective PLL's output division factor.
13:15 PLLIDF	PLL Input Division Factor. The value of this field drives the input division factor of the PLL embedded in the device. Refer to the PLL specifications to define the exact mapping between the PLLIDF value and the respective PLL's input division factor.
25:31 PLLLDF	PLL Loop Division Factor. The value of this field drives the loop division factor of the PLL embedded in the device. Refer to the PLL specifications to define the exact mapping between the PLLLDF value and the respective PLL's loop division factor.

76.5.6 STCU2 Watchdog Register Granularity (STCU_WDG)

The STCU_WDG register defines the time budget for MBIST execution, providing protection against dead-locks or infinite loop conditions during the self-test procedure. When the offline self-test sequence is not run and the STCU_RUN[BYP] bit is not set, the bits 15 to 0 define the timeout before the STCU_ERR[WDTO] bit is set and the STCU2 core clock is switched off.

When the offline or online self-test sequence is run, it defines the timeout of the execution run.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active
- Online self-test phase is active and STCU_CFG[WRP] = 0

Offset: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WDGEOC															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDGEOC															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1567. STCU2 Watchdog Register Granularity (STCU_WDG)

Table 1530. STCU_WDG field descriptions

Field	Description
0:31 WDGEOC	Watchdog End of Count Timer. This value must be set in order to define the time budget respective to the offline and online self-test execution and check that everything is working correctly in this timeslot. The allowed time delays are: 0x00000000 1*k*STCU2 core clock cycles 0x00000001 2*k*STCU2 core clock cycles 0x00000002 3*k*STCU2 core clock cycles … 0xFFFFFFFD 4294967294*k*STCU2 core clock cycles 0xFFFFFFFE 4294967295*k*STCU2 core clock cycles 0xFFFFFFFF 4294967296*k*16 STCU2 core clock cycles where k is: – ‘1’ for offline self-test – ‘16’ for online self-test **Note:** STCU2 core clock is specified through STCU_CFG[CLK_CFG] field.

76.5.7 STCU2 Interrupt Flag Register (STCU_INT_FLG)

The STCU_INT_FLG register includes the MBIST interrupt pending bit. This bit is effective only during the online self-test phase and managed only when the respective control bit in the STCU_RUNSW register (MBIE) is enabled.

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MBIFLG	0
W															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1568. STCU2 Interrupt Flag Register (STCU_INT_FLG)

Table 1531. STCU_INT_FLG field descriptions

Field	Description
30 MBIFLG	MBIST Interrupt Flag. 0 No interrupt is pending. 1 An interrupt highlighting that the online execution of the concurrent scheduled MBIST is completed is pending. Write 1 to clear the bit.

76.5.8 STCU2 CRC Expected Status Register (STCU_CRCE)

The STCU_CRCE register includes the expected signature of the CRC-32 (ethernet protocol). The CRC-32 bit engine is initialized at STCU_CRCE_RES and computes the CRC signature of the STCU2 critical signals. When STCU_CFG[CRCEN] = 1, it provides the self check capability of the STCU2 managing the CRCS[SW] bits of the STCU_ERR_STAT register. The polynomial CRC used to evaluate the status of this register is the following:

CRC-32 (ethernet protocol)

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions are true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Note: The expected value of the CRC depends on multiple internal critical signals and can differ among different samples.

Offset: 0x001C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CRCE															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CRCE															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1569. STCU2 CRC Expected Status Register (STCU_CRCE)

Table 1532. STCU_CRCE field descriptions

Field	Description
0:31 CRCE	CRC expected signature

76.5.9 STCU2 CRC Read Status Register (STCU_CRCR)

The STCU_CRCR register reports the value obtained at the end of the offline and online self-test sequence. It may be used for diagnoses and also for additional checking with respect to the CRCS[SW] bit of the STCU_ERR_STAT register.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0020 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRCR															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCR															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1570. STCU2 CRC Read Status Register (STCU_CRCR)

Table 1533. STCU_CRCR field descriptions

Field	Description
0:31 CRCR	Read CRC signature

76.5.10 STCU2 Error Register (STCU_ERR_STAT)

The STCU_ERR_STAT register includes the status flags respective to the STCU2 internal error conditions that occurred during configuration or the online and offline self-test execution.

The Unrecoverable Faults Status Flag (UFSF) and Recoverable Faults Status Flag (RFSF) can be set and cleared using the FCCU dedicated channels as described in *Section 76.6.4: Design implementation information*.

Access to this register is described in *Figure 1571* and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality is implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set and clean capability.

Offset: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	ABORTHW[1]	0	0	0	0	LOCKESW[1]	WDTOSW[1]	CRCSSW[1]	ENGESW[1]	INVPSW[1]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	UFSF	RFSF	0	0	0	LOCKE	WDTO	CRCS	ENGE	INVP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can always read this field. The content of this field is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 1571. STCU2 Error Register (STCU_ERR_STAT)

Table 1534. STCU_ERR_STAT field descriptions

Field	Description
6 ABORTHW	Online hardware abort flag. 0 No hardware abort was requested during the online self-test sequence 1 A hardware abort was detected during the online self-test sequence
11 LOCKESW	Online LOCK Error. 0 If PLL is enabled, it is correctly locked during the self-test sequence 1 When the PLL is enabled, this flag highlights an unexpected PLL unlock event during the online self-test sequence execution. The online self-test run is stopped and the status of the current running MBISTs is saved in the respective registers. The LOCK signal is monitored during the MBIST run when STCU_RUNSW[MBSWPLLEN] = 1.
12 WDTOSW	Online Watchdog timeout. 0 MBIST time slots have completed within the assigned watchdog time. 1 MBIST time slots have not completed in the assigned watchdog time or there are internal mismatches in the end of execution signals. The condition that flags the failures is the following: – MBIST BEND status flags that at least one of the selected MBIST run is not finished
13 CRCSSW	Online CRC Status. This flag is activated *only* when online self-test is performed and STCU_CFG[CRCEN] is set to 1, otherwise it is always forced to 0. This prevents the generation of the respective STCU_ERR_STAT[UFSF/RFSF] bits. In both the cases, the content of the STCU_CRCR register reports the current evaluated CRC. 0 Successful CRC comparison or comparison has been masked (STCU_CFG[CRCEN] set to 0) 1 Failed CRC comparison

Table 1534. STCU_ERR_STAT field descriptions (continued)

Field	Description
14 ENGESW	Online Engine Error. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: – STCU2 state machines (watchdog, master and loader/shifter) selects an unused code – Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned Serial bend status of the selected MBIST and the global BEND flag are not aligned
15 INVPSW	Online Invalid Pointer. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: – Initial MBIST pointer is out of range – Error in MBIST linking (execution generates an infinite loop)
22 UFSF	Unrecoverable Faults Status Flag This flag reports the global status of the UF. 0 No errors that trigger the unrecoverable faults condition 1 There are errors that trigger the unrecoverable faults condition
23 RFSF	Recoverable Faults Status Flag 0 No errors that trigger the recoverable faults condition 1 There are errors that trigger the recoverable faults condition
27 LOCKE	Offline LOCK Error 0 The PLL is correctly locked during the self-test sequence 1 When the PLL is enabled, this flag highlights an unexpected PLL unlock event during the offline self-test sequence execution. The online self-test run is stopped and the status of the current running MBISTs is saved in the respective registers. The LOCK signal is monitored during the MBIST run when STCU_RUN[MBPLLEN] = 1.
28 WDTO	Offline Watchdog timeout. 0 MBIST time slots have been completed within the assigned watchdog time. 1 MBIST time slots have not been completed within the assigned watchdog time or there are internal mismatches among end of execution signals. The conditions that flag the failures are the following: – The STCU_RUN[RUN] bit or the STCU_RUN[BYP] bit are not set before the watchdog reach the end of count – MBIST BEND status flags that at least one of the selected MBIST run is not finished
29 CRCS	Offline CRC status. This flag is activated *only* when offline self-test is performed STCU_CFG[CRCEN] is set to 1, otherwise it is always forced to 0. This prevents the generation of the respective STCU_ERR_STAT[UFSF/RFSF] bits. In both the cases, the content of the STCU_CRCR register reports the current evaluated CRC. 0 Successful CRC comparison or comparison has been masked (STCU_CFG[CRCEN] programmed to 0) 1 Failed CRC comparison

Table 1534. STCU_ERR_STAT field descriptions (continued)

Field	Description
30 ENGE	Offline Engine Error. 0 Valid engine execution 1 Invalid engine execution. The following conditions set this bit: – STCU2 state machines (watchdog, master and loader/shifter) selects an unused code – Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned – Serial bend status of the selected MBIST and the global BEND flag are not aligned
31 INVP	Offline Invalid pointer. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: – Initial MBIST pointer is out of range – Error in MBIST linking (execution generates an infinite loop)

76.5.11 STCU2 Error FM Register (STCU_ERR_FM)

The STCU_ERR_FM register defines the fault mapping of the STCU2 faults described in the STCU_ERR_STAT register in terms of Unrecoverable or recoverable faults. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions are true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LOCKEUFM	WDTOUFM	CRCSUFM	ENGEUFM	INVPUFM
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1572. STCU2 Error FM Register (STCU_ERR_FM)

Table 1535. STCU_ERR_FM field descriptions

Field	Description
27 LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
28 WDTOUFM	WatchDog timeout Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
29 CRCSUFM	CRC Status Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
30 ENGEUFM	Engine Error Unrecoverable fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
31 INVPUFM	Invalid Pointer Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable mapping

76.5.12 STCU2 Offline MBIST 1 Status Register (STCU_MBS1)

The STCU_MBS1 register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

Offset: 0x0044 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS31(1)	MBS30(1)	MBS29(1)	MBS28(1)	MBS27(1)	MBS26(1)	MBS25(1)	MBS24(1)	MBS23(1)	MBS22(1)	MBS21(1)	MBS20(1)	MBS19(1)	MBS18(1)	MBS17(1)	MBS16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS15(1)	MBS14(1)	MBS13(1)	MBS12(1)	MBS11(1)	MBS10(1)	MBS9(1)	MBS8(1)	MBS7(1)	MBS6(1)	MBS5(1)	MBS4(1)	MBS3(1)	MBS2(1)	MBS1(1)	MBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1573. STCU2 Offline MBIST 1 Status Register (STCU_MBS1)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to Table 93: MBIST registers in STCU2 with available fields in Device configuration chapter.

Table 1536. STCU_MBS1 field descriptions

Field	Description
0:31 MBS[31:0]	Offline status (NMCUT range = 0 to 31) of the selected MBIST **Note:** This bit is meaningful when the respective bit of the STCU_MBE1 register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

76.5.13 STCU2 Offline MBIST 2 Status Register (STCU_MBS2)

The STCU_MBS2 register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

Offset: 0x0048 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS63(1)	MBS62(1)	MBS61(1)	MBS60(1)	MBS59(1)	MBS58(1)	MBS57(1)	MBS56(1)	MBS55(1)	MBS54(1)	MBS53(1)	MBS52(1)	MBS51(1)	MBS50(1)	MBS49(1)	MBS48(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS47(1)	MBS46(1)	MBS45(1)	MBS44(1)	MBS43(1)	MBS42(1)	MBS41(1)	MBS40(1)	MBS39(1)	MBS38(1)	MBS37(1)	MBS36(1)	MBS35(1)	MBS34(1)	MBS33(1)	MBS32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1574. STCU2 Offline MBIST 2 Status Register (STCU_MBS2)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1537. STCU_MBS2 field descriptions

Field	Description
0:31 MBS[63:32]	MBS*x*: Offline status (NMCUT range = 32 to 63) of the selected MBIST **Note:** This bit is meaningful when the respective bit of the STCU_MBE2 register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

76.5.14 STCU2 Offline MBIST 1 End Flag Register (STCU_MBE1)

The STCU_MBE1 register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

Offset: 0x0054 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31[1]	MBE30[1]	MBE29[1]	MBE28[1]	MBE27[1]	MBE26[1]	MBE25[1]	MBE24[1]	MBE23[1]	MBE22[1]	MBE21[1]	MBE20[1]	MBE19[1]	MBE18[1]	MBE17[1]	MBE16[1]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15[1]	MBE14[1]	MBE13[1]	MBE12[1]	MBE11[1]	MBE10[1]	MBE9[1]	MBE8[1]	MBE7[1]	MBE6[1]	MBE5[1]	MBE4[1]	MBE3[1]	MBE2[1]	MBE1[1]	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1575. STCU2 Offline MBIST 1 End Flag Register (STCU_MBE1)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1538. STCU_MBE1 field descriptions

Field	Description
0:31 MBE[31:0]	Offline End status (0 to 31) of the selected MBIST. 0 MBIST execution still ongoing 1 MBIST execution finished

76.5.15 STCU2 Offline MBIST 2 End Flag Register (STCU_MBE2)

The STCU_MBE2 register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

Offset: 0x0058 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE63(1)	MBE62(1)	MBE61(1)	MBE60(1)	MBE59(1)	MBE58(1)	MBE57(1)	MBE56(1)	MBE55(1)	MBE54(1)	MBE53(1)	MBE52(1)	MBE51(1)	MBE50(1)	MBE49(1)	MBE48(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE47(1)	MBE46(1)	MBE45(1)	MBE44(1)	MBE43(1)	MBE42(1)	MBE41(1)	MBE40(1)	MBE39(1)	MBE38(1)	MBE37(1)	MBE36(1)	MBE35(1)	MBE34(1)	MBE33(1)	MBE32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1576. STCU2 Offline MBIST 2 End Flag Register (STCU_MBE2)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1539. STCU_MBE2 field descriptions

Field	Description
0:31 MBE[63:32]	Offline End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

76.5.16 STCU2 Online MBIST 1 Status Register (STCU_MBS1SW)

The STCU_MBS1SW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

Offset: 0x0064 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW31(1)	MBSSW30(1)	MBSSW29(1)	MBSSW28(1)	MBSSW27(1)	MBSSW26(1)	MBSSW25(1)	MBSSW24(1)	MBSSW23(1)	MBSSW22(1)	MBSSW21(1)	MBSSW20(1)	MBSSW19(1)	MBSSW18(1)	MBSSW17(1)	MBSSW16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15(1)	MBSSW14(1)	MBSSW13(1)	MBSSW12(1)	MBSSW11(1)	MBSSW10(1)	MBSSW9(1)	MBSSW8(1)	MBSSW7(1)	MBSSW6(1)	MBSSW5(1)	MBSSW4(1)	MBSSW3(1)	MBSSW2(1)	MBSSW1(1)	MBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1577. STCU2 Online MBIST 1 Status Register (STCU_MBS1SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1540. STCU_MBS1SW field descriptions

Field	Description
0:31 MBSSW[31:0]	Online status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

76.5.17 STCU2 Online MBIST 2 Status Register (STCU_MBS2SW)

The STCU_MBS2SW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

Offset: 0x0068 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW63(1)	MBSSW62(1)	MBSSW61(1)	MBSSW60(1)	MBSSW59(1)	MBSSW58(1)	MBSSW57(1)	MBSSW56(1)	MBSSW55(1)	MBSSW54(1)	MBSSW53(1)	MBSSW52(1)	MBSSW51(1)	MBSSW50(1)	MBSSW49(1)	MBSSW48(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW47(1)	MBSSW46(1)	MBSSW45(1)	MBSSW44(1)	MBSSW43(1)	MBSSW42(1)	MBSSW41(1)	MBSSW40(1)	MBSSW39(1)	MBSSW38(1)	MBSSW37(1)	MBSSW36(1)	MBSSW35(1)	MBSSW34(1)	MBSSW33(1)	MBSSW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1578. STCU2 Online MBIST 2 Status Register (STCU_MBS2SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1541. STCU_MBS2SW field descriptions

Field	Description
0:31 MBSSW[63:32]	MBSSW*x*: online status (NMCUT range = 32 to 63) of the selected MBIST. 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

76.5.18 STCU2 Online MBIST 1 End Flag Register (STCU_MBE1SW)

The STCU_MBE1SW register includes the end flag respective to the execution of the selected online MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

Offset: 0x0074 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW31(1)	MBESW30(1)	MBESW29(1)	MBESW28(1)	MBESW27(1)	MBESW26(1)	MBESW25(1)	MBESW24(1)	MBESW23(1)	MBESW22(1)	MBESW21(1)	MBESW20(1)	MBESW19(1)	MBESW18(1)	MBESW17(1)	MBESW16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW15(1)	MBESW14(1)	MBESW13(1)	MBESW12(1)	MBESW11(1)	MBESW10(1)	MBESW9(1)	MBESW8(1)	MBESW7(1)	MBESW6(1)	MBESW5(1)	MBESW4(1)	MBESW3(1)	MBESW2(1)	MBESW1(1)	MBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1579. STCU2 Online MBIST 1 End Flag Register (STCU_MBE1SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1542. STCU_MBE1SW field descriptions

Field	Description
0:31 MBESW[31:0]	MBESWx: online End status (0 to 31) of the selected MBIST. 0 MBIST execution still ongoing 1 MBIST execution finished

76.5.19 STCU2 Online MBIST 2 End Flag Register (STCU_MBE2SW)

The STCU_MBE2SW register includes the end flag respective to the execution of the selected online MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

Offset: 0x0078 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW63(1)	MBESW62(1)	MBESW61(1)	MBESW60(1)	MBESW59(1)	MBESW58(1)	MBESW57(1)	MBESW56(1)	MBESW55(1)	MBESW54(1)	MBESW53(1)	MBESW52(1)	MBESW51(1)	MBESW50(1)	MBESW49(1)	MBESW48(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW47(1)	MBESW46(1)	MBESW45(1)	MBESW44(1)	MBESW43(1)	MBESW42(1)	MBESW41(1)	MBESW40(1)	MBESW39(1)	MBESW38(1)	MBESW37(1)	MBESW36(1)	MBESW35(1)	MBESW34(1)	MBESW33(1)	MBESW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1580. STCU2 Online MBIST 2 End Flag Register (STCU_MBE2SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1543. STCU_MBE2SW field descriptions

Field	Description
0:31 MBESW[63:32]	Online End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

76.5.20 STCU2 MBIST 1 Unrecoverable FM Register (STCU_MBUFM1)

The STCU_MBUFM1 register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0084 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM31(1)	MBUFM30(1)	MBUFM29(1)	MBUFM28(1)	MBUFM27(1)	MBUFM26(1)	MBUFM25(1)	MBUFM24(1)	MBUFM23(1)	MBUFM22(1)	MBUFM21(1)	MBUFM20(1)	MBUFM19(1)	MBUFM18(1)	MBUFM17(1)	MBUFM16(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM15(1)	MBUFM14(1)	MBUFM13(1)	MBUFM12(1)	MBUFM11(1)	MBUFM10(1)	MBUFM9(1)	MBUFM8(1)	MBUFM7(1)	MBUFM6(1)	MBUFM5(1)	MBUFM4(1)	MBUFM3(1)	MBUFM2(1)	MBUFM1(1)	MBUFM0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1581. STCU2 MBIST 1 Unrecoverable FM Register (STCU_MBUFM1)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1544. STCU_MBUFM1 field descriptions

Field	Description
0:31 MBUFM[31:0]	MBIST Unrecoverable Fault Mapping. 0 Recoverable fault mapping 1 Unrecoverable fault mapping

76.5.21 STCU2 MBIST 2 Unrecoverable FM Register (STCU_MBUFM2)

The STCU_MBUFM2 register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.5: STCU2 configuration in Device configuration chapter for the number of NMCUT available in this device.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0088 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM63(1)	MBUFM62(1)	MBUFM61(1)	MBUFM60(1)	MBUFM59(1)	MBUFM58(1)	MBUFM57(1)	MBUFM56(1)	MBUFM55(1)	MBUFM54(1)	MBUFM53(1)	MBUFM52(1)	MBUFM51(1)	MBUFM50(1)	MBUFM49(1)	MBUFM48(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM47(1)	MBUFM46(1)	MBUFM45(1)	MBUFM44(1)	MBUFM43(1)	MBUFM42(1)	MBUFM41(1)	MBUFM40(1)	MBUFM39(1)	MBUFM38(1)	MBUFM37(1)	MBUFM36(1)	MBUFM35(1)	MBUFM34(1)	MBUFM33(1)	MBUFM32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1582. STCU2 MBIST 2 Unrecoverable FM Register (STCU_MBUFM2)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 93: MBIST registers in STCU2 with available fields* in Device configuration chapter.

Table 1545. STCU_MBUFM2 field descriptions

Field	Description
0:31 MBUFM[63:32]	MBIST Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping

76.5.22 STCU2 MBIST Control *k* Register (STCU_MB_CTRL*k*)

The STCU_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0600 + k*0x4 (k = 0 to 34) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTR								CSM	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1583. STCU2 MBIST Control *k* Register (STCU_MB_CTRL*k*)

Table 1546. STCU_MB_CTRL*k* field descriptions

Field	Description
0:7 PTR	Next MBIST pointer PTR defines the logical pointer to the next MBIST to be scheduled. The next MBIST is scheduled concurrently to the current one if the CSM bit is set to 1, otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer, the CSM bit has to be set sequential (0) to define this is the end of the list. The self-test procedure is stopped when the current MBIST has been completed. PTR = 0x10 + $MBIST_i$ (Pointer to MBIST) for i = 0 to XX: please refer to the MBIST partitions table for the $MBIST_i$ code. PTR = 0xFF: Pointer to NIL. No additional MBIST execution. Any other value is considered as invalid pointer and an error is set in the STCU_ERR register.
8 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode

76.6 Use cases and limitations

The STCU2 module is designed to give high flexibility during the online and offline self-test. This section shows the programming operations that have to be performed to correctly use this module in three typical use cases:

- Offline self-test sequence after a reset trigger event
- Bypass mode of the self-test sequence after a reset trigger event
- Online self-test sequence

The STCU setup has to be defined based on the application-specific safety, boot time and consumption requirements. Contact ST representative for support.

76.6.1 Offline self-test sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and MBIST execution parameters stored in the NVM memory. The target is to cover the amount of physical defects in the digital logic and in the system RAM or ROM according to the system specifications. The sequence is the following:

- Unlock the offline STCU2 access writing the Key1/Key2 sequence in the STCU_SKC register.
- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU_MBUFM1/2 registers.
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU_ERR_FM register.
- Program the STCU_MB_CTRL registers of each NMCUT to be executed.
- Program the STCU_WDG register to define the time budget assigned for MBIST execution.
- Program the STCU_CFG register to define:
  - the core and MBIST TCK clock prescaling factor setting the CLK_CFG bits,
  - the PMOSEN or CHKBRD to define the algorithm to be used during MBIST run,
  - the CRCREN bit to enable the self-checking feature,
  - set the pointer to the first NMCUT to be executed.
- If the internal CRC comparison has been enabled, program the CRCE value expected at the end of the offline self-test sequence in the STCU_CRCE register.
- If PLL usage IS enabled, program the STCU_PLL_CFG value to set the required system frequency.
- Program the RUN bit in the STCU_RUN register to enable the offline self-test execution and the LMPLLEN bits to enable the PLL during MBIST execution according to STCU_PLL_CFG register content.
- Start the offline self-test depending on PLL management and MBIST execution order.
- Wait the WDGEOC execution run time specified in the STCU_WDG register.
- The STCU2 switches-off the core clock at the end of the self-test run and releases the PLL control if this feature is enabled.
- The STCU2 resets the STCU_RUN[RUN] bit and the device exits the boot sequence. If there are failures, the STCU2 flags these failures to the FCCU by raising one or both of the stcu_ncf and stcu_cf lines.
- The software application reads the STCU_MBS1/2 flag registers, depending on the number of memories, to check the failed RAM or ROM memory BIST when UF or RF are detected.
- The software application reads the STCU_MBE1/2 flag registers, depending on the number of memories, to check the RAM or ROM memory BIST still running when UF or RF are detected.
- The software application reads the bit INV, ENGE, CRCS, WDTO, LOCKE of the STCU_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF are detected.
- If the CRCEN bit is enabled, the software application reads the CRCE and CRCR registers to check the coherence with the bit CRCS of the STCU_ERR register.

Note: The software application read operations on the STCU2 CRC register after offline self-test execution has successfully completed are an additional safety layer added to improve the already implemented auto self-test feature included in STCU2.

76.6.2 Online self-test Sequence

This is the typical mode of using the STCU2 module during the application run. The IPS interface is used to program the STCU2 registers and to schedule the MBIST execution. Since the online operations clean up the content of the RAM (MBIST), activation of this mode requires caution. Follow this sequence:

- Unlock the online STCU2 access writing the Key1/Key2 sequence in the STCU_SKC register.
- Check and clear the STCU_CFG[WRP] bit to open the IPS access on the STCU2 online self-test registers.
- The MBIST and STCU2 internal fault reaction should already be programmed during the offline self-test sequence. If they have not been programmed or there is a need to change them, they can be overwritten by setting the registers: STCU_MBUFM1/2 and STCU_ERR_FM.
- Overwrite/Program the STCU_MB_CTRL registers of each NMCUT to be executed.
- Overwrite/Program the STCU_WDG register to define the time budget assigned for MBIST execution.
- Overwrite/Program the STCU_CFG register to define:
  - the core and MBIST TCK clock prescaling factor setting the CLK_CFG bits,
  - the PMOSEN or CHKBRD to define the algorithm to be used during MBIST run,
  - the CRCREN bit to enable the self-checking feature
  - set the pointer to the first NMCUT to be executed.
- If the internal CRC comparison is enabled, overwrite/program the CRCE value expected at the end of the online self-test sequence in the STCU_CRCE register.
- Program the STCU_RUNSW[RUNSW] bit to enable the online self-test execution, the MBIE to enable the Interrupt requests, and the MBSWPLLEN bits to enable the PLL lock signal monitor during MBIST execution.
- Start the online self-test depending on PLL management and MBIST execution order.
- Wait the WDGEOC execution run time specified in the STCU_WDG register.
- If the MBIE is enabled, an interrupt is generated at the end of all the MBIST runs. The software application has to manage these interrupt signals and clean up the respective flag (MBIFLG) in the STCU_INT_FLG register.
- The STCU2 switches off the core clock at the end of the self-test run.
- The STCU2 resets the STCU_RUNSW[RUNSW] bit. In both the reset cases and in case of errors, the STCU2 flags the failure to the FCCU by rising one or both of the stcu_ncf and stcu_cf lines.
- The software application reads the STCU_MBS1SW/2SW flag registers, depending on the number of memories, to check the failed RAM/ROM memory BIST when UFs or RFs are detected.
- The software application reads the STCU_MBE1SW/2SW flag registers, depending on the number of memories, to check the RAM/ROM memory BIST is still running when UFs or RFs are detected.

- The software application reads the bit INVPSW, ENGESW, CRCSSW, WDTOSW, LOCKESW of the STCU_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UFs or RFs are detected.
- If the CRCEN bit is enabled, the software application reads the CRCE and CRCR registers to check the coherence with the STCU_ERR[CRCSSW] bit.

Note: The software application read operations on the STCU2 CRC register after offline self-test execution has successfully completed are an additional safety layer added to improve the already implemented auto self-test feature included in STCU2.

76.6.3 Bypass USER Mode

In this mode, the user application parameters stored in Flash and read by SSCM are written to skip the self-test procedure after a reset trigger event is applied. It might be useful if the device is not configured for Safety applications. Follow this sequence:

1. Unlock the STCU2 access writing the offline Key1/Key2 sequence in the STCU_SKC register.
2. Set the BYP bit in the STCU_RUN register.

After the BYP bit is set, the core clock is switched off, the self-test sequence is not applied and the system can wake up and start the user application.

76.6.4 Design implementation information

The following list reports the limitations regarding the current implementation:

- Every NMCUT MBIST can be run one time during the on/offline self-test procedure.
- As the software has full control of the system while applications are running during the online self-test, the STCU2 may not take control of the PLL, but only monitor the lock signal to flag incorrect unlock events.

77 Register Protection (REG_PROT)

77.1 Introduction

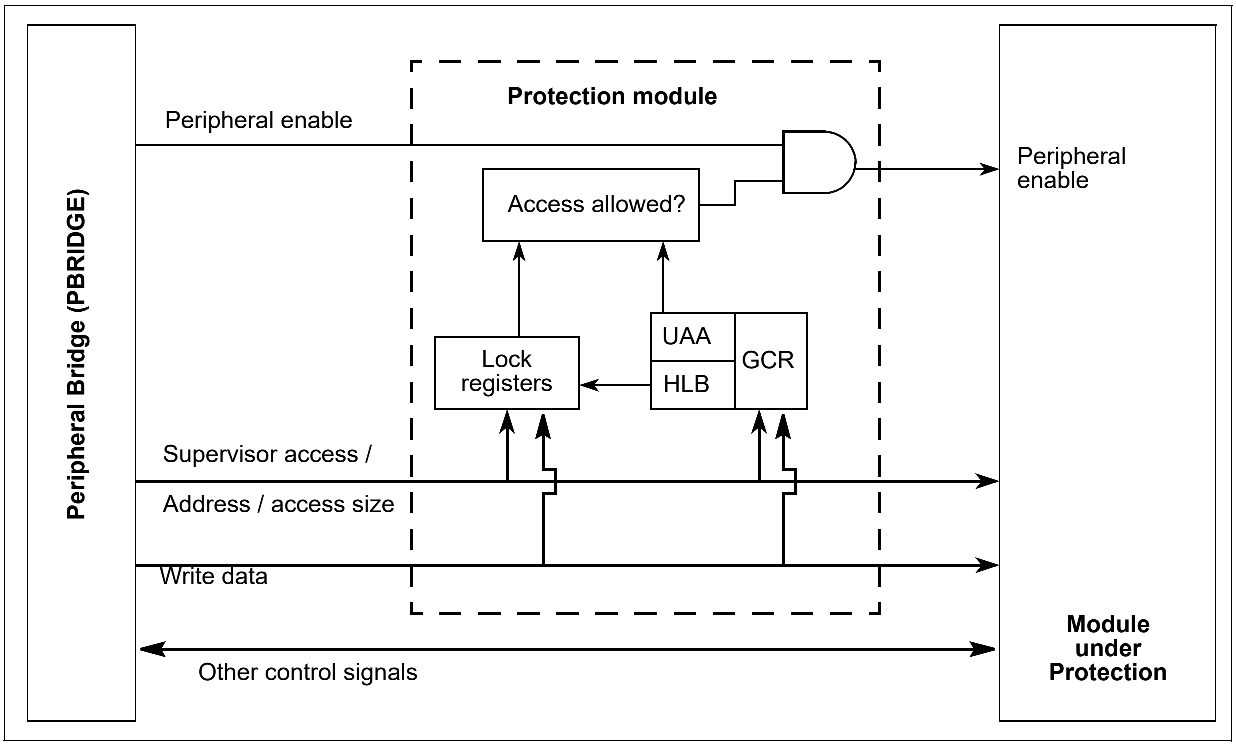
77.1.1 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

Note: Refer to the Device configuration chapter for protected registers.

The protection module is located between the module under protection and the peripheral bridge (PBRIDGE). This is shown in *Figure 1584*.

Figure 1584. REG_PROT block diagram



77.1.2 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

Also, hard lock bit protection can only be cleared by a system reset once set.

77.1.3 Modes of operation

The Register Protection module is operable when the module under protection is operable.

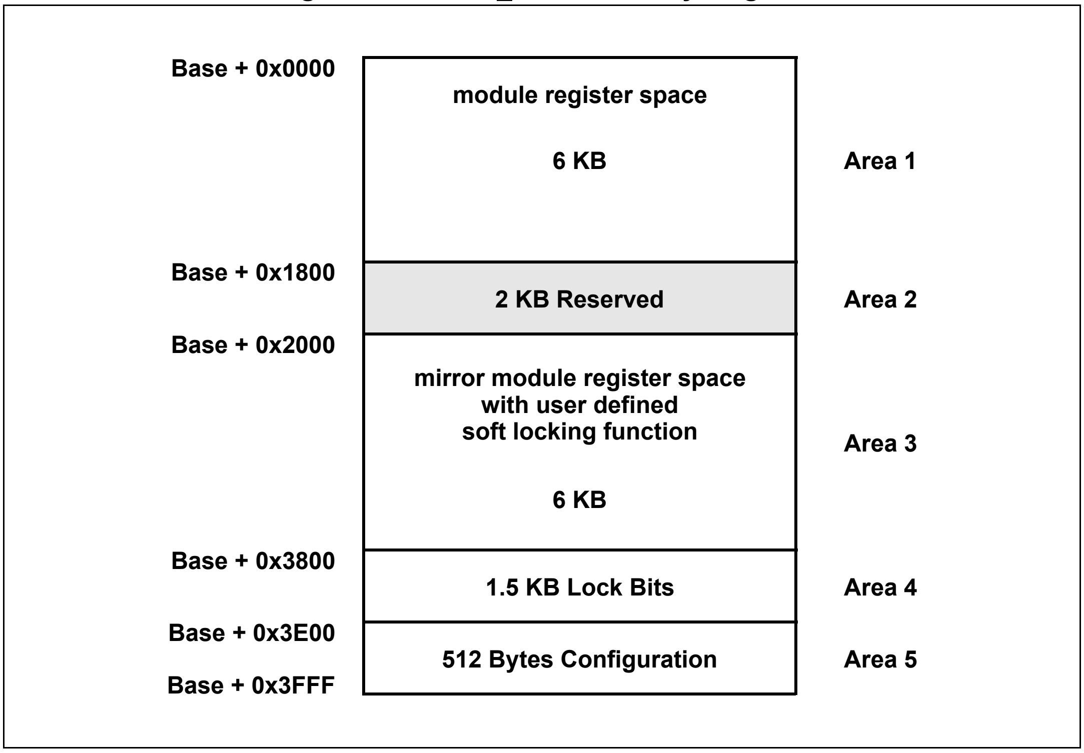
77.2 External signal description

There are no external signals.

77.3 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in *Figure 1585*.

Figure 1585. REG_PROT memory diagram



- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an atomic write and lock operation. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.

- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

77.3.1 Memory map

Table 1547 shows the memory map for a module which is protected via a REG_PROT module.

Table 1547. REG_PROT memory map

Offset	Use	Location
0x0000	Module Register 0 (MR0)	*Section 77.3.2.1*
0x0001	Module Register 1 (MR1)	*Section 77.3.2.1*
0x0002	Module Register 2 (MR2)	*Section 77.3.2.1*
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)	*Section 77.3.2.1*
0x1800 - 0x1FFF	Reserved	
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)	*Section 77.3.2.2*
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)	*Section 77.3.2.2*
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)	*Section 77.3.2.2*
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3	*Section 77.3.2.3*
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7	*Section 77.3.2.3*
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 - Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143	*Section 77.3.2.3*
0x3E00 - 0x3FFB	Reserved	
0x3FFC	Global Configuration Register (GCR)	*Section 77.3.2.4*

Note: Reserved registers in area #1 will be handled as specified in the protected module's documentation.

77.3.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

77.3.2.1 Module registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

77.3.2.2 Module register and Set Soft Lock Bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBR*n*.SLB*m*, according to the mapping described in *Table 1549*.

77.3.2.3 Soft Lock Bit Register (SLBR0-1535)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each Soft Lock Bit also has a corresponding Write Enable bit in the same register that controls whether the Soft Lock Bit can be written. The following table shows the mapping between the Soft Lock Bits to the bytes in memory area

Table 1548. Soft lock bits vs. protected address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

Note: As many as 1536 Soft Lock Bit Registers (SLBRn) may be implemented, depending on the number of protected module register bytes.

Access in User Mode is read-only; in Supervisor Mode access is read/write.

Offset: 0x3800-0x3DFF

Access: Read always
Supervisor write

	0	1	2	3	4	5	6	7
R	0	0	0	0	SLB0	SLB1	SLB2	SLB3
W	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

Figure 1586. Soft Lock Bit Register (SLBR*n*)

Table 1549. SLBR*n* field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB): WE0 enables writing to SLB0 1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB): WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified
2 WE2	Write Enable Bits for Soft Lock Bits (SLB): WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB): WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MR*n* register: SLB0 can block accesses to MR[*n* *4 + 0] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
5 SLB1	Soft Lock Bits for one MR*n* register: SLB1 can block accesses to MR[*n* *4 + 1] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
6 SLB2	Soft Lock Bits for one MR*n* register: SLB2 can block accesses to MR[*n* *4 + 2] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
7 SLB3	Soft Lock Bits for one MR*n* register: SLB3 can block accesses to MR[*n* *4 + 3] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable

77.3.2.4 Global Configuration Register (GCR)

This register controls the configuration of the locking of the SLB's and the level of user access allowed to the protected registers.

Note: Access in User Mode is read-only; in Supervisor Mode access is read/write.

Offset: 0x3FFC

Access: Read Always
Supervisor write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HLB	0	0	0	0	0	0	0	UAA	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1587. Global Configuration Register (GCR)

Table 1550. GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit. This register cannot be cleared once it is set by software. It can only be cleared by a system reset. 1 All SLB bits are write protected and cannot be modified 0 All SLB bits are accessible and can be modified.
8 UAA	User Access Allowed. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.

Note: The GCR.UAA bit has no effect on the allowed access modes for the registers in the REG_PROT module.

77.4 Functional description

The following sections describe the functional characteristics of the Register Protection module.

77.4.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size is device and module specific, please refer to *Chapter 7: Device configuration for details.*

For all addresses that are protected there are SLBR*n*.SLB*m* bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBR*n*.SLB*m* bit is always 0b0 no matter what software is writing to.

77.4.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBR*n*.SLB*m* bit needs to be changed. This can be done using the following methods:

- Modify the SLBR*n*.SLB*m* directly by writing to area #4
- Set the SLBR*n*.SLB*m* bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

77.4.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBR*n*.SLB*m* bit has a mask bit SLBR*n*.WE*m* which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

Figure 1588 shows two modification examples. In the left example there is a write access to the SLBR*n* register specifying a mask value which allows modification of all SLBR*n*.SLB*m* bits. The example on the right specifies a mask which only allows modification of the bits SLBR*n*.SLB[3:1].

Figure 1588. Change lock settings directly via Area #4

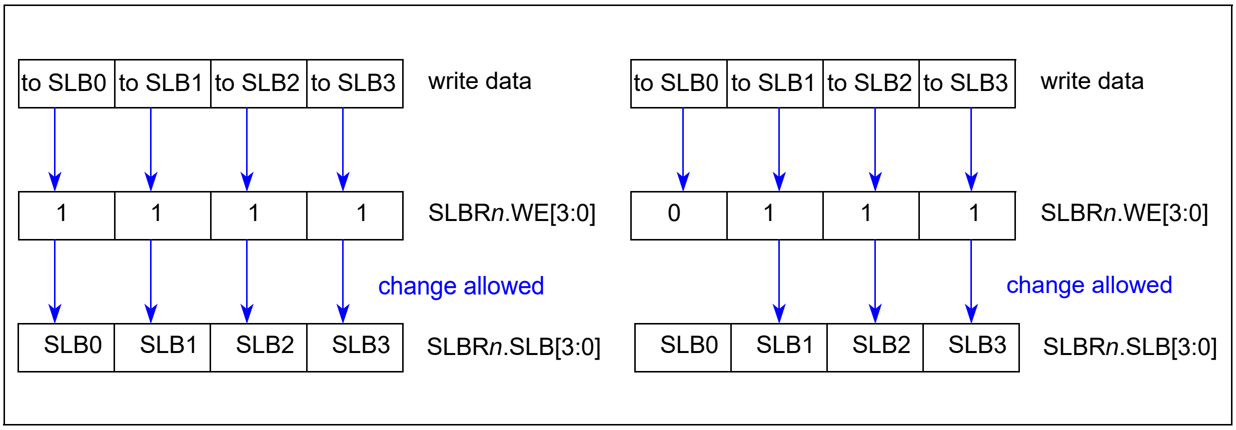
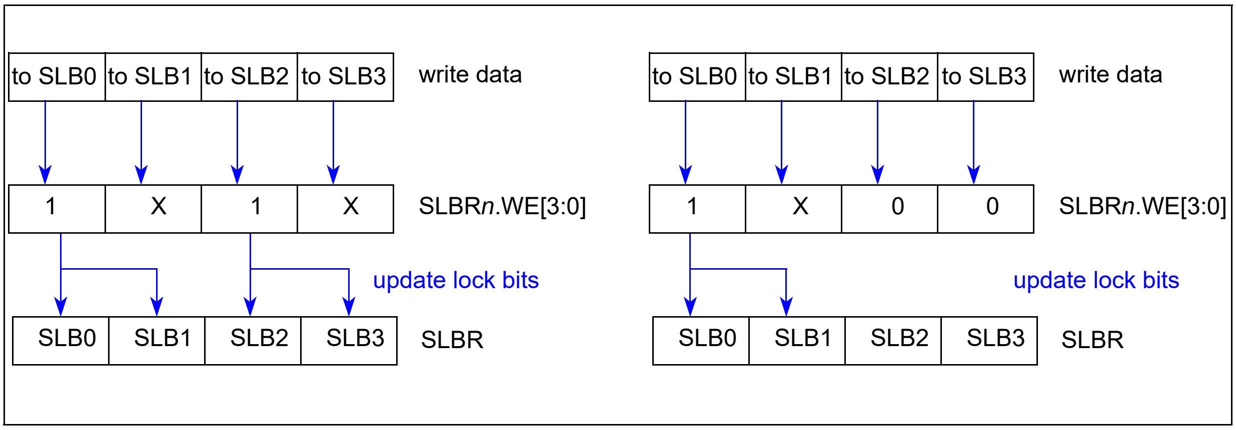


Figure 1588 shows four registers that can be protected 8-bit wise. In *Figure 1589* registers with 16-bit protection and in *Figure 1590* registers with 32-bit protection are shown:

Figure 1589. Change lock settings for 16-bit protected addresses

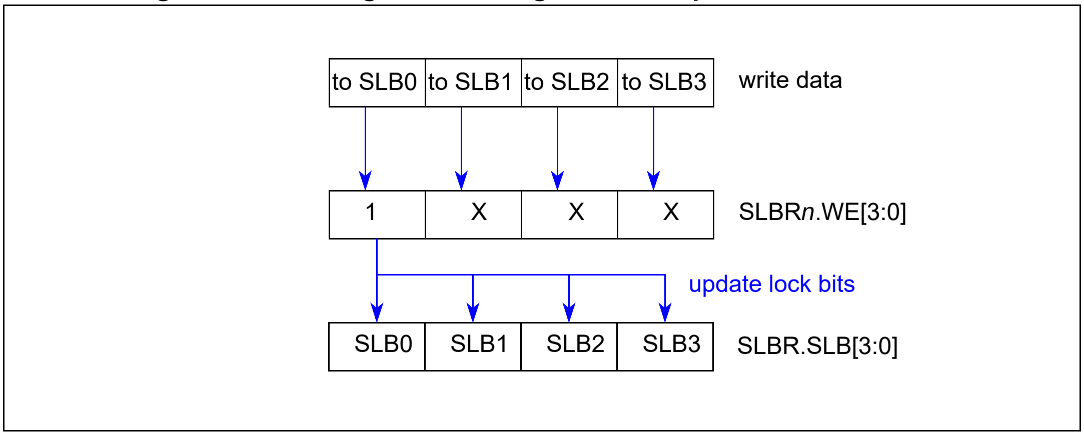


On the right side of *Figure 1589* it is shown that the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[1] also. This is done as the address reflected by SLBR*n*.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBR*n*.WE[0] must be set while SLBR*n*.WE[1] does not matter. As the enable bits SLBR*n*.WE[3:2] are cleared the lock bits SLBR*n*.SLB[3:2] remain unchanged.

In the example on the left side of *Figure 1589* the data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] and the data written to SLBR*n*.SLB[2] is mirrored to SLBR*n*.SLB[3] as for both registers the write enables are set.

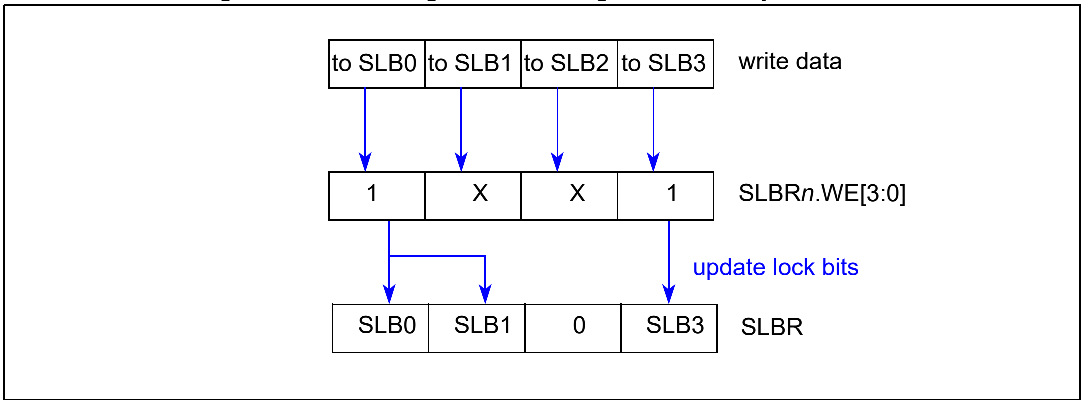
In *Figure 1590* a 32-bit wise protected register is shown. When SLBR*n*.WE[0] is set the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[3:1] also. Otherwise SLBR*n*.SLB[3:0] remains unchanged.

Figure 1590. Change lock settings for 32-bit protected addresses



In *Figure 1591* an example is shown which has a mixed protection size configuration:

Figure 1591. Change lock settings for mixed protection

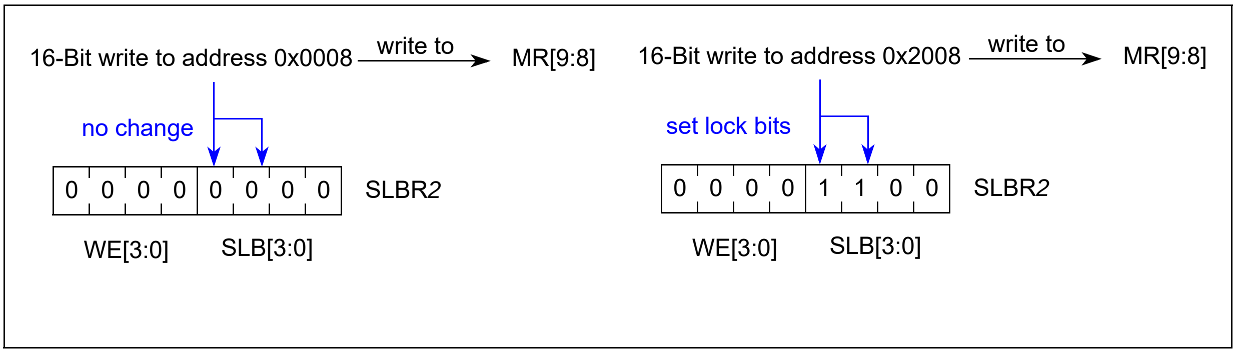


The data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBR*n*.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBR*n*.SLB[3] is written to SLBR*n*.SLB[3].

77.4.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. *Figure 1592* shows one example:

Figure 1592. Enable Locking Via Mirror Module Space (Area #3)

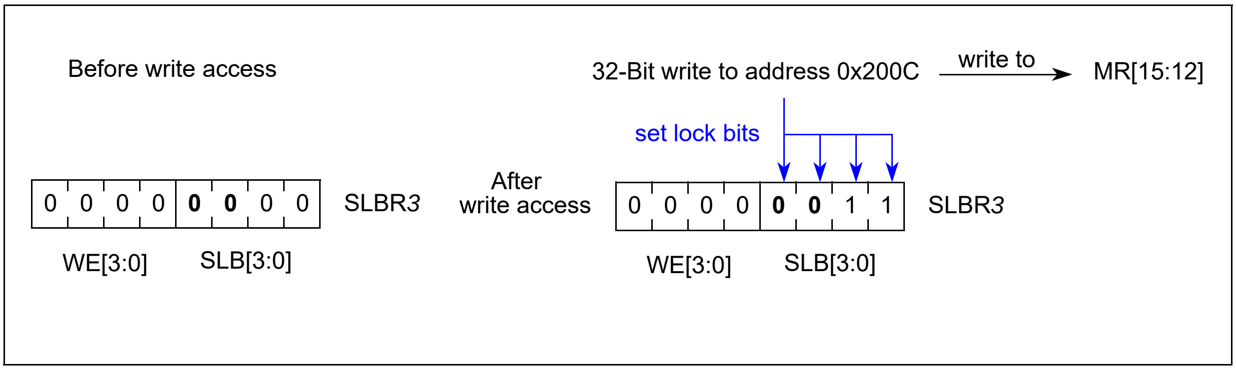


When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of *Figure 1589*).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of *Figure 1589*).

Figure 1593 shows an example where some addresses are protected and some are not:

Figure 1593. Enable locking for protected and unprotected addresses



In the example in *Figure 1593* addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note: Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

77.4.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in *Section 77.4.2.1: Change lock settings directly via Area #4* and *Section 77.4.2.2: Enable locking via mirror module space (Area #3)* is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there is a system reset.

77.4.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to *Figure 1585*.

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set.

77.5 Initialization/application information

77.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See *Section 77.3.2*). In summary, after reset, locking for all MR*n* registers is disabled. The registers can be accessed in Supervisor Mode only.

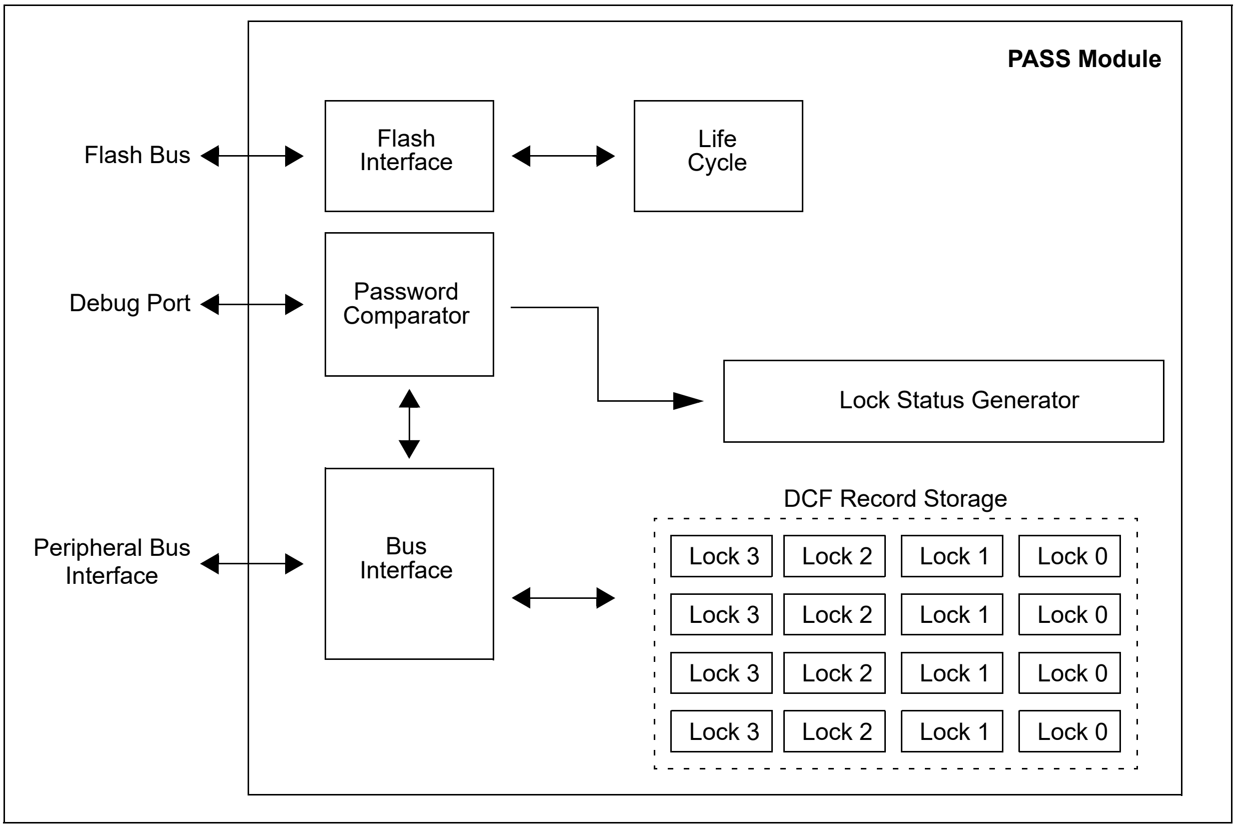
78 Password and Device Security Module (PASS)

78.1 Introduction

78.1.1 Overview

The Password and Device Security (PASS) module shown in *Figure 1594* receives password challenges and determines their validity. It also maintains the device security and access states.

Figure 1594. PASS module block diagram



78.1.2 Features

The PASS includes these distinctive features:

- Lifecycle status
- Password comparison
- JTAG password comparison
- Fuse Bypass Configuration and Password Comparison

78.1.3 Modes of operation

The PASS operates identically in all system modes.

78.2 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the PASS. (DCF clients are described in *Section 78.3: DCF clients*.)

Table 1551 shows the memory map for the PASS. Note that all addresses are offsets: the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section.

Table 1551. PASS memory map

Offset	Register	Access	Reset Value	DCF Client Address(1) (ADDR[14:0])	Section
0x0000	Lifecycle Status Register (LCSTAT)	R	0xX0000007	None	*Section 78.2.1.1*
0x0008	Challenge Selector Register (CHSEL)	R/W	0x00000000	None	*Section 78.2.1.2*
0x0010	Challenge Status Register (CSTAT)	R	0x00000000	None	*Section 78.2.1.3*
0x0014-0x001F	Reserved				
0x0020	Challenge Input Register 0 (CIN0)	W	0x00000000	None	*Section 78.2.1.4*
0x0024	Challenge Input Register 1 (CIN1)	W	0x00000000	None	*Section 78.2.1.4*
0x0028	Challenge Input Register 2 (CIN2)	W	0x00000000	None	*Section 78.2.1.4*
0x002C	Challenge Input Register 3 (CIN3)	W	0x00000000	None	*Section 78.2.1.4*
0x0030	Challenge Input Register 4 (CIN4)	W	0x00000000	None	*Section 78.2.1.4*
0x0034	Challenge Input Register 5 (CIN5)	W	0x00000000	None	*Section 78.2.1.4*
0x0038	Challenge Input Register 6 (CIN6)	W	0x00000000	None	*Section 78.2.1.4*
0x003C	Challenge Input Register 7 (CIN7)	W	0x00000000	None	*Section 78.2.1.4*
0x0040-0x00AF	Reserved				
None	Censorship	None	0x00000000	0x00B0	*Section 78.3.1*
0x00C4-0x00DF	Reserved				
None	Fuse Bypass	None	0x00000001	0x00E0	*Section 78.3.2*
0x00E4-0x00FF	Reserved				
Password Group 0					
0x0100	Lock 0 Status Register (LOCK0_PG0)	R/W	0xXXXXXXXX(2)	0x0100	*Section 78.2.1.5*
0x0104	Lock 1 Status Register (LOCK1_PG0)	R/W	0x0000XXXX(2)	0x0104	*Section 78.2.1.6*
0x0108	Lock 2 Status Register (LOCK2_PG0)	R/W	0xXXXXXXXX(2)	0x0108	*Section 78.2.1.7*
0x010C	Lock 3 Status Register (LOCK3_PG0)	R/W	0xXFXXXXXX(2)	0x010C	*Section 78.2.1.8*

Table 1551. PASS memory map (continued)

Offset	Register	Access	Reset Value	DCF Client Address(1) (ADDR[14:0])	Section
Password Group 1					
0x0110	Lock 0 Status Register (LOCK0_PG1)	R/W	0xXXXXXXXX(2)	0x0110	Section 78.2.1.5
0x0114	Lock 1 Status Register (LOCK1_PG1)	R/W	0x0000XXXX(2)	0x0114	Section 78.2.1.6
0x0118	Lock 2 Status Register (LOCK2_PG1)	R/W	0xXXXXXXXX(2)	0x0118	Section 78.2.1.7
0x011C	Lock 3 Status Register (LOCK3_PG1)	R/W	0xXFXXXXXX(2)	0x011C	Section 78.2.1.8
Password Group 2					
0x0120	Lock 0 Status Register (LOCK0_PG2)	R/W	0xXXXXXXXX(2)	0x0120	Section 78.2.1.5
0x0124	Lock 1 Status Register (LOCK1_PG2)	R/W	0x0000XXXX(2)	0x0124	Section 78.2.1.6
0x0128	Lock 2 Status Register (LOCK2_PG2)	R/W	0xXXXXXXXX(2)	0x0128	Section 78.2.1.7
0x012C	Lock 3 Status Register (LOCK3_PG2)	R/W	0xXFXXXXXX(2)	0x012C	Section 78.2.1.8
Password Group 3					
0x0130	Lock 0 Status Register (LOCK0_PG3)	R/W	0xXXXXXXXX(2)	0x0130	Section 78.2.1.5
0x0134	Lock 1 Status Register (LOCK1_PG3)	R/W	0x0000XXXX(2)	0x0134	Section 78.2.1.6
0x0138	Lock 2 Status Register (LOCK2_PG3)	R/W	0xXXXXXXXX(2)	0x0138	Section 78.2.1.7
0x013C	Lock 3 Status Register (LOCK3_PG3)	R/W	0xXFXXXXXX(2)	0x013C	Section 78.2.1.8

1. DCF Client Address column shows what is the DCF used to preload the register value during reset. If the column value is “None”, it means that the register value at reset is fixed in hardware.
2. Reset value depends on password group lock state.

Note: Non-implemented registers generate bus aborts if enabled by the SOC.

78.2.1 Register descriptions

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write ‘0’ to these bits; they always read ‘0’.

78.2.1.1 Lifecycle Status register (LCSTAT)

The Lifecycle Status Register reflects the production status and censorship status of the device. It is possible to mature the device (move the device lifecycle forward), but never to revert the lifecycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external lifecycle management module (like SSCM). It is a 3-bit lifecycle. This register simply displays the lifecycle input from external blocks.

Offset: 0x0000 Access: Software Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	FBE	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	x(1)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	LIFE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

1. Reset value depends on the Censorship DCF client.

Figure 1595. Lifecycle Status register (LCSTAT)

PASS_LCSTAT register functions are as shown in *Table 1552*.

Table 1552. LCSTAT register field descriptions

Field	Description
0 CNS	This field indicates the state of the Censorship DCF Client. The bit can be set (or cleared) via DCF Record in UTEST Flash. Refer to *Section 78.5.3: Censoring and uncensoring the device*. 0 Device is uncensored 1 Device is censored
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to 1, otherwise it is 0. Unsuccessful transmission clears this bit. **Note:** Refer to the SPC584Bx Microcontroller Security Reference Manual for details.
2 FBE	Fuse Bypass enable This field indicates whether the Fuse Bypass DCF Client is set to Fuse Bypass Enable. 0 No Fuse Bypass 1 Fuse Bypass Enable **Note:** Refer to *Section 78.3.2* for details.
29:31 LIFE	The lifecycle setting is manipulated externally (SSCM). Pass displays a 3-bit input lifecycle. 000 Failure Analysis mode 001 Reserved 010 OEM Production 011 Customer Delivery 110 ST Production 111 In Field

78.2.1.2 Challenge Selector register (CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

When a master accesses this register, the PASS module stores the number of the master in the LOCK3_PGn register of the selected passgroup and sets the PGL bit in that register.

The ID of the accessing master sets the CMST field of the Challenge Status register.

Offset: 0x0008 Access: Software Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GRP	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1596. CHSEL register

CHSEL register functions are as shown in *Table 1553*.

Table 1553. CHSEL register field descriptions

Field	Description
30:31 GRP	Password Group to unlock 00 Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

78.2.1.3 Challenge Status register (CSTAT)

This register provides status information for the password challenge.

Offset: 0x0010 Access: Software Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CMST			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1597. Challenge Status register (CSTAT)

CSTAT register functions are as shown in *Table 1554*.

Table 1554. CSTAT register field descriptions

Field	Description
28:31 CMST	ID of the master which has last written the CHSEL register

Only the master shown in this register can access the CINn registers; access by other masters result in a comparison fail regardless of the values written.

78.2.1.4 Challenge Input register (CIN*n*)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

Valid password criteria is that it should not be all Zeros or all Ones

Offset: 0x0020 + *n**0x4 (*n* = 0 to 7) Access: Software Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	PW32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	PW32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1598. Challenge Input register (CIN*n*)

CIN register functions are as shown in *Table 1555*.

Table 1555. CIN*n* register field descriptions

Field	Description
0:31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7—in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been executed.

All accesses to these registers need to be made by the bus master which wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

78.2.1.5 Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the lifecycle has matured to *OEM Production* or older.

Offset: 0x0100 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	TSLOCK	ATSL	LOWLOCK													
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R / W	MIDLOCK															
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

1. If no DCF records have been written to change the lock state, the bit will reset to ‘1’.

Figure 1599. Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

LOCK0_PG*n* register functions are as shown in *Table 1556*.

Table 1556. LOCK0_PG*n* register field descriptions

Field	Description
0 TSLOCK	UTEST NVM Lock This bit is used to lock the UTEST NVM block from programs (erase protection not needed since UTEST NVM is OTP and not erasable).
1 ATSL	Alternate Interface UTEST NVM Lock. This bit is used to lock the UTEST NVM block from programs on the alternate interface (erase protection not needed since UTEST NVM is OTP and not erasable).
2:15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16:31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is ‘1’, indicating that a block is locked.

Warning: When Password Slot Access bits are set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to In-Field renders the part unusable.
When Password Slot Access bits are not set, failure to program valid passwords or DCF records for all password

groups before lifecycle matures to OEM Production renders the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

78.2.1.6 Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the lifecycle matures to *OEM Production* or older.

Offset: 0x0104 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	HIGHLOCK															
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

1. If no DCF records have been written to change the lock state, the bit will reset to ‘1’.

Figure 1600. Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

LOCK1_PG*n* register functions are as shown in *Table 1557*.

Table 1557. LOCK1_PG*n* register field descriptions

Field	Description
16:31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written, is ‘1’, which indicates a block is locked.

Warning: When Password Slot Access bits are set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to In-Field renders the part unusable.
When Password Slot Access bits are not set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to OEM Production renders the part unusable.
However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

78.2.1.7 Password Group *n* - Lock 2 status register (LOCK2_PG*n*)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the lifecycle is matured to *OEM Production* or older.

Offset: 0x0108 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	A256KLOCK															
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	A256KLOCK															
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

1. If no DCF records have been written to change the lock state, the bit resets to ‘1’.

Figure 1601. Password Group *n* - Lock 2 status register (LOCK2_PG*n*)

Note: Lock registers can only be accessed with 32-bit operations.

LOCK2_PG*n* register functions are as shown in *Table 1558*.

Table 1558. LOCK2_PG*n* register field descriptions

Field	Description
0:31 A256KLOCK	256K address space block lock A256KLOCK[31:0] A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is ‘1’, which indicates a block is locked.

Warning: **When Password Slot Access bits are set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to In-Field renders the part unusable.**
When Password Slot Access bits are not set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to OEM Production renders the part unusable.
However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase are possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

78.2.1.8 Password Group *n* - Lock 3 status register (LOCK3_PG*n*)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the PGL bit is cleared and the Lock registers become writable.

The LOCK bits do not come into effect before the lifecycle matures to *OEM Production* or older.

Offset: 0x010C + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	RL4[1]	RL3[1]	RL2[1]	RL1[1]	RL0[1]
W																
Reset	x[2]	x[2]	x[2]	0	1	1	1	1	0	0	0	x[2]	x[2]	x[2]	x[2]	x[2]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KLOCK[1]															
W																
Reset	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]	x[3]

1. Not all devices include this field. Refer to the device reference manual’s chip configuration chapter for details.
2. Reset value depends on lifecycle. If lifecycle is *Customer delivery* or earlier, the value will be 0, otherwise 1.
3. If no DCF records have been written to change the lock state, the bit will reset to ‘1’.

Figure 1602. Password Group *n* - Lock 3 status register (LOCK3_PG*n*)

LOCK3_PG*n* register functions are as shown in *Table 1559*.

Table 1559. LOCK3_PG*n* register field descriptions

Field	Description
0 PGL	Password Group Lock This bit may be set in software to lock the password group, when the device is older than *Customer Delivery*. Before that the value of the bit is always 0. This bit is only updatable from SW path. It cannot be updated from DCF loading from Flash. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally (if lifecycle is *Customer Delivery* or earlier, this bit has no effect and the registers are always unlocked).
1 DBL	Debug Interface Lock The default value of this bit would be ‘1’ if not updated from flash. 0 Debug interface is unlocked. Accessed to all JTAG clients is allowed (Except HSM). 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked.
2 MO	Master Only 0 All masters can modify the passgroup settings if PGL is cleared. 1 Only the Master which unlocked the passgroup can change the passgroup settings.
4:7 MSTR	Master Access Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.

Table 1559. LOCK3_PG*n* register field descriptions (continued)

Field	Description
11 RL4	Read Lock Region 4 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 4 is not protected. 1 Region 4 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
12 RL3	Read Lock Region 3 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 3 is not protected. 1 Region 3 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
13 RL2	Read Lock Region 2 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 2 is not protected. 1 Region 2 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
14 RL1	Read Lock Region 1 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 1 is not protected. 1 Region 1 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
15 RL0	Read Lock Region 0 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 0 is not protected. 1 Region 0 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
16:31 A256KLOCK	256K address space block lock A256KLOCK[47:32] A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is ‘1’, which indicates a block is locked.

Warning: When Password Slot Access bits are set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to In-Field renders the part unusable.
When Password Slot Access bits are not set, failure to program valid passwords or DCF records for all password

groups before lifecycle matures to OEM Production renders the part unusable.
However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase are possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

For operation of the region read protect feature, refer to *Section 78.4.3: Flash Memory Read Protection*.

Once a censored device enters the *Failure Analysis* lifecycle, read protection is active regardless whether the system is in debug mode or not. It is still possible to unlock Flash regions with the correct passwords, however. Flash UTest region controlled by LOCK3[0]is read unlocked in Failure Analysis lifecycle.

78.3 DCF clients

Where a DCF client is listed, the reset value of that register may automatically be preloaded during reset based on the data stored in DCF Records in Flash UTEST region.

The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section.

78.3.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically preloaded during reset with data stored in DCF Record format in the Flash UTEST region. The Censorship DCF client defines whether the MCU is censored or not (Refer to *Table 1551*).

DCF Client Address: 0x00B0 Censorship

Access: DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not defined.
2. This value is loaded by DCF Record during reset.

Figure 1603. Censorship register (not accessible)

Table 1560. Censorship register field descriptions

Field	Description
16:31	Censorship If this value is 0x55AA the MCU is NOT censored. If this value is anything other that 0x55AA, the MCU is censored.

78.3.2 Fuse Bypass DCF Client

This DCF client is used to configure the password protected Fuse bypass feature. Refer to *Section 78.4.5: Fuse Bypass* for details.

The Fuse Bypass DCF Client is not directly readable in software, but the state of the Fuse Bypass bit can be read in the LCSTAT register (refer to *Lifecycle Status register (LCSTAT)*).

The DCF client is written during Destructive Reset Phase 3 and compared during Functional Reset Phase 3; if this compare does not match the data already written, the logic will decode No fuse bypass.

DCF Client Address: 0x00E0 Fuse Bypass

Access: DCF Record Write Once Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																FB(2)
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	1

1. This bit is not defined.
2. This value is loaded by DCF Record during reset.

Figure 1604. Fuse Bypass DCF Client

Table 1561. Fuse Bypass register field descriptions

Field	Description
31 FB	Fuse Bypass enable 0 The Fuse can NOT be Bypassed 1 Bypass of the fuse is possible with a password

78.3.3 Password Slot Access

Password Slot Access is a write-once DCF client. The DCF controls access for both read and write starting from Lifecycle = OEM Production. If the bit is 1, the slot is still accessible when Life Cycle = OEM Production (and blocked starting from Life Cycle = In Field). Otherwise if the bit is set to 0, the slot is blocked starting from Life Cycle = OEM Production.

DCF Client Address: 0x00B4 Password Slot Access

Access: DCF Record Write Once

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R													Password Group 3 Access	Password Group 2 Access	Password Group 1 Access	Password Group 0 Access
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1605. Password Slot Access DCF Client

Table 1562. Password Slot Access DCF Client field descriptions

Field	Description
28:31	Password Slot Access per password group

78.4 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

78.4.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access; and flash memory read protection.

The MCU can be Censored by writing a DCF Record to flash that writes to the Censorship DCF client (refer to *Section 78.3.1: Censorship*).

78.4.2 Debug Interface Access

The debug interface in fully enabled in Customer_Delivery Lifecycle. But from OEM_Production lifecycle, it can be blocked using censorship and selectively by either entering a 256 bits password via JTAG or by software clearing the DBL bit the PASS_LOCK3 register (refer to *78.2.1.8: Password Group n - Lock 3 status register (LOCK3_PGn)*). Both of these debug enabling options are further conditioned by lifecycle.

The following table describes all options for enabling/blocking Debug Interface access.

Table 1563. Debug Interface Enable Truth table

<table>
<tr><th colspan="4">Input Conditions</th><th>Outcome</th></tr>
<tr><th>Lifecycle</th><th>Censorship</th><th>Debug Enable (LOCK3[DBL] bit)</th><th>JTAG Password</th><th>Debug Interface Enable</th></tr>
<tr><td>Customer Delivery</td><td>Don't Care</td><td>Don't Care</td><td>Don't Care</td><td>Enabled</td></tr>
<tr><td rowspan="4">OEM Production</td><td>Not censored</td><td>Don't Care</td><td>Don't Care</td><td>Enabled</td></tr>
<tr><td rowspan="3">Censored</td><td>Unlocked (DBL=0)</td><td>Don't Care</td><td>Enabled</td></tr>
<tr><td rowspan="2">Locked (DBL=1)</td><td>Matched</td><td>Enabled</td></tr>
<tr><td>Not Matched</td><td>Blocked</td></tr>
<tr><td rowspan="4">In Field</td><td>Not censored</td><td>Don't Care</td><td>Don't Care</td><td>Enabled</td></tr>
<tr><td rowspan="3">Censored</td><td>Unlocked (DBL=0)</td><td>Don't Care</td><td>Enabled</td></tr>
<tr><td rowspan="2">Locked (DBL=1)</td><td>Matched</td><td>Enabled</td></tr>
<tr><td>Not Matched</td><td>Blocked</td></tr>
<tr><td>Failure Analysis</td><td>Don't Care</td><td>Don't Care</td><td>Don't Care</td><td>Enabled</td></tr>
</table>

The HSM can also block Debug Interface Access, even if it enabled by PASS. This allows HSM to granted Debug Interface Access after an encrypted secure challenge/response protocol with an external tool (refer to the SPC584Bx Microcontroller Security Reference Manual).

78.4.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge register in the JTAGC, can be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled (according to the table above) until the next destructive reset.

78.4.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Lifecycle, Censorship and whether the Debug Interface access is enabled or blocked. *Table 1564: Flash Memory Read Protection truth table* describes the combination of conditions that drive Flash Memory Read Protection.

The intention of the feature is to protect the contents of the flash from being read. The flash memory is readable in normal operation, allowing the application to run, but becomes read protected when a Debug tool is attached and enabled (JTAG password or LOCK3[DBL] bit).

The Flash memory is also read protected when the lifecycle status is *Failure_Analysis*. This lifecycle is used for a module field failure, where the device is returned for failure analysis. This protection prevents the contents of the flash memory from being read.

The flash memory is divided into a number of Regions (refer to *Chapter 7: Device configuration*, *Section 7.11.1: Password and Device Security Module (PASS) configuration*).

Table 1564. Flash Memory Read Protection truth table

INPUTS				OUTPUTS	
Lifecycle	**Censorship**	**Debug Interface Enable from HSM**	**Lock3[RLx]**	**Read Protected RL[0] (UTEST)**	**Read Protected RL[4:1] (Other Regions)**
STMicroelectronics Production / Customer Delivery	Don't Care	Don't Care	Don't Care	Readable	Readable
OEM Production / IN Field	Uncensored	Don't Care	Don't Care	Readable	Readable
	Censored	Blocked	Don't Care	Readable	Readable
		Enabled	Locked (1)	Read Blocked	Read Blocked
			Unlocked (0)	Readable	Readable
Failure Analysis	Uncensored	Don't Care	Don't Care	Readable	Readable

78.4.4 UTEST protection

During the *Customer Delivery* lifecycle, the user must program a valid password or a DCF record that writes LOCKm_PGn registers to unprotect the UTest for each password group in order to prevent permanent and irreversible locking of Flash blocks.

78.4.5 Fuse Bypass

78.4.5.1 Fuse

Some devices include an additional security feature, whereby a silicon fuse is used to gate access to Flash Test Mode when the device is in Failure_Analysis lifecycle. The fuse must be blown, creating physical damage to the silicon, before access is granted to Flash Test Mode. The PASS implements a configurable feature to bypass the requirement to blow the fuse to enter flash test mode, by entering a password instead. This feature is called Fuse Bypass.

78.4.5.1.1 Use Case: No Fuse Bypass

Customers who want no alternative to the Fuse to enter Flash Test Mode in FA lifecycle:

- During customer assembly, customer programs DCF Record in UTEST to write this DCF client with 0.
- And as a back-up, during customer assembly, the customer could program the Fuse Bypass Password to a random value.

78.4.5.1.2 Use Case: Fuse Bypass Password

Customers who require a password to enter Flash Test Mode in FA lifecycle, but do not require the added sophistication of blowing a fuse:

- During customer assembly, customer programs DCF Record in UTEST to write this DCF client with 1.
- During customer assembly, the customer programs the Fuse Bypass Password into Flash.
- The customer maintains a system for retrieving these passwords for the life of the module.
- If the part is returned to STMicroelectronics for FA, the customer must provide the Fuse Bypass password that was originally programmed into the device.
- If the password is lost or irretrievable, the customer must accept that FA can only be performed if the die is removed from the package and the fuse blown.

78.4.5.1.3 Use Case: No Fuse. No Password

Customers who are less concerned with security and will accept neither fuse nor password to enter Flash Test Mode in FA lifecycle:

- Customer need do nothing special for this mode in customer assembly.
- When the part is returned to STMicroelectronics for FA, The DCF Client is still in the default state of 1. This is Fuse Bypass.
- The Fuse Bypass password should still be in the erased state (all 1). The tester can enter all 1 as the password to gain access to Flash Test Mode.

78.4.5.2 Fuse Bypass Password

The Fuse Bypass Password is a password entered by the STMicroelectronics via JTAG interface, during device FA, to gain access to Flash Test Mode. Flash Test Mode is required to perform Vt Distribution analysis and other tests on an MCU. This password is only functional during FA lifecycle. Flash blocks that are assigned as sealed during production are locked against any form of Flash test, including reading, erasing and Vt Distribution analysis.

78.4.5.2.1 Fuse Bypass Password Generation

The Fuse Bypass password is created by the customer during their module assembly process. It must be programmed into UTest Flash while in CUSTOMER_DELIVERY lifecycle.

When an MCU is returned to STMicroelectronics for FA, the MCU must be in FA lifecycle and this password must also be provided to STMicroelectronics to allow access to Flash Test Mode without blowing the fuse. Therefore the customer should develop a process for managing Fuse Bypass passwords for the life of the module.

If the password is not available when a device is returned for FA, STMicroelectronics must remove the die from the package and blow the fuse before entry is possible to Flash Test Mode and Flash FA can be undertaken.

78.4.5.2.2 Fuse Bypass Password storage

The Fuse Bypass Password is 224 bits (28 bytes) wide and is to be stored in UTest Flash at a fixed location. Refer to the device reference manual for the address of the Fuse Bypass password.

The password region of UTest Flash is readable only when in CUSTOMER_DELIVERY lifecycle. In any lifecycle after this (including FA lifecycle), this region is read protected.

78.4.5.2.3 Fuse Bypass Password Challenge entry

The Fuse Bypass password challenge is entered by an external tool to the password register in JTAGC.

As the Fuse Bypass Password is 224 bits; and the JTAGC password register is 256, only the least significant 224 bits of the register are used.

78.4.5.2.4 Fuse Bypass Password compare

The Fuse Bypass Password challenged, is compared against the password stored in UTest Flash. Only the least significant 224 bits are compared.

If the password matches, Flash Test Mode access is allowed until the next POR reset.

There are no passwords that the hardware will detect as illegal. All 0 and all 1 are valid passwords.

78.5 Initialization/application information

78.5.1 Reset

The reset state of each individual bit is shown within *Section 78.2.1: Register descriptions*. However, many bits receive their values based on DCF records stored in the UTest Flash memory block.

78.5.2 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTEST Flash, apart from the available space for DCF records. It is possible to set and clear them multiple times.

Table 1565. CS/ADDR values for Lock registers

Register	DCF CS/ADDR
PW Group 0	
Lock0	0xXXXX0100(1)
Lock1	0xXXXX0104(1)
Lock2	0xXXXX0108(1)
Lock3	0xXXXX010C(1)
PW Group 1	
Lock0	0xXXXX0110(1)

Table 1565. CS/ADDR values for Lock registers (continued)

Register	DCF CS/ADDR
Lock1	0xXXXX0114(1)
Lock2	0xXXXX0118(1)
Lock3	0xXXXX011C(1)
PW Group 2	
Lock0	0xXXXX0120(1)
Lock1	0xXXXX0124(1)
Lock2	0xXXXX0128(1)
Lock3	0xXXXX012C(1)
PW Group3	
Lock0	0xXXXX0130(1)
Lock1	0xXXXX0134(1)
Lock2	0xXXXX0138(1)
Lock3	0xXXXX013C(1)

1. Value depends on the DCF Client Select assigned to the PASS module in the device.

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF and other fields to 0, the DCF record has to be added as shown in *Table 1566*.

Table 1566. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA55AF	Start Record
0x04	0x00000000	
0x08	0x000000FF	PW Group 1, Lock0 Set MIDLOCK to 0xFF
0x0C	0xXXXX0110(1)	
0x10	0xFFFFFFFF	Stop Record
0x14	0xFFFFFFFF	
...	...	

1. Value depends on the DCF Client Select assigned to the PASS module in the device.

78.5.3 Censoring and uncensoring the device

There is no restriction on setting and clearing the censorship status in the UTest Flash, apart from the available space for DCF records. It is possible to censor and un-censor multiple times.

The DCF Client Select / Address for the Censorship DCF client, is a concatenation of the DCF Client Select (assigned outside of PASS) and the address 0x00B0. Therefore 0xXXXX00B0. In order to un-censor the device, the data field of the record needs to be written with 0x----55AA (upper 16 bits are not compared). Any other value censors the device.

On delivery the device is censored.

79 Tamper Detection Module (TDM)

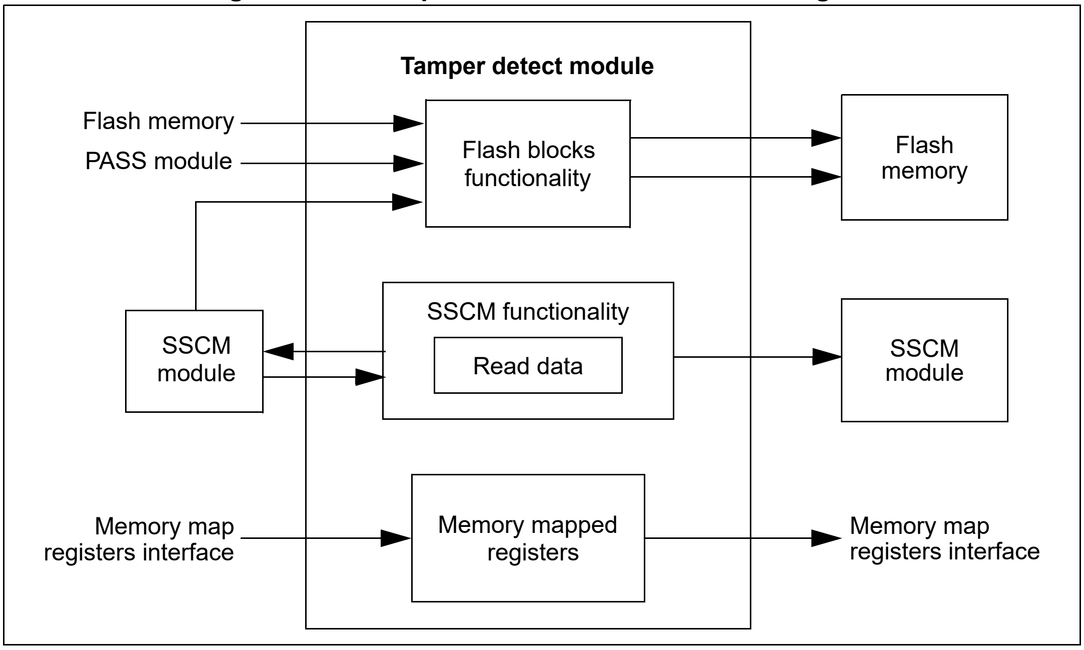
79.1 Overview

The Tamper Detection Module provides a type of flash memory erase protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. The mechanism does not *prevent* an erase operation on any block within a TDR: it requires a record to be written before the operation can execute. Collectively, the records are referred to as a “diary”. At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 6 TDRs can be defined via DCF records.

Figure 1606 shows the block diagram of the TDM and its interface to other system components.

Figure 1606. Tamper Detection Module block diagram



Security information is passed to the TDM by the SSCM module. Among the data are:

- OTP (One-Time Programmable) settings for each of the flash blocks
- Base address for the diary in the flash
- Block assignments for the TDRs
- Tamper region override information (via the DCF bus)
- Software tamper region override

The TDM module controls the flash blocks for erase operations and sets them as OTP

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are already erased can be programmed. Over-programming is not possible.

Note: The PASS module provides the control for erase operation for each flash block as well as for each region of the flash memory, and the PASS settings have priority over the settings of the TDM.

The flash block indicates to the TDM the address of the programmed word. This address is internally decoded and provides a means to verify whether the block is part of any TDRs. The memory map registers interface provides a means to the user to read information of the TDM registers.

79.2 Features

The Tamper Detection Module supports these distinctive features:

- Erase counter (diary) for each TDR to record up to 256 write attempts on the flash blocks
  - The diary is divided into 6 parts (TDRs), each consisting of 256 128-bit records
  - The diary block should be assigned as OTP region in flash
  - The diary area can store any type of relevant information
- Tamper Detection Region Block Assignment
  - Any flash block can be assigned to any TDR and be controlled
- 32-bit Erase Lock DCF clients for each flash memory TDR defined
- 32-bit OTP Enable DCF clients to disable the erase operation for an entire flash block
- Override TDR via DCF records[y]
  - Allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation
- Memory Map Registers Interface for reading via software of specific registers

79.3 External signal description

There are no external signals driving or being driven by the TDM.

79.4 DCF client

TDM-related DCF records perform the following functions:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip flash memory program/erase activity
- Permanently disable one or more Tamper Regions
- Permanently disable ability for software to permanently override one or more TDRs
- Configure flash memory as One Time Programmable (OTP) on a per-block basis

The following table gives an overview on the TDM DCF clients implemented.

---

y. The Tamper Region Override DCF record is a “write one only” DCF record. If an override is applied to a TDR it is permanent and cannot be reversed.

Table 1567. DCF client address map

DCF Client Name	Description	Strategy
Diary Base Address	Stores the base address of the tamper detection diary	write-once, write 1 to 0 only
Tamper Region Override	Stores information of which TDR is to be overridden	write 0 to 1 only
OTPEN[nb] (OTP Enable)	Stores information of which flash blocks is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	write 0 to 1 only
TDR[n]_LOCK[m] Region Assignment	Stores information of which flash block is assigned to a TDR – "*n*" specifies a TDR – "*m*" specifies the flash blocks as per the LOCK registers of the flash memory	write 0 to 1 only

79.4.1 Diary Base Address (DBA) DCF client

This DCF client holds base address information of the diary. *Figure 1607* shows the diagram for this register.

Caution: This DCF client is implemented as Write Once and is writable from 1 to 0 only. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0000 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	DBA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	DBA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1607. Diary Base Address (DBA) DCF client

Table 1568. DBA field descriptions

Field	Description
0:31 DBA	Diary Base Address This field holds the base address of the diary. This information is then used to verify if the diary being programmed matches any TDR region. **Note:** Diary Base Address must be on a 16KB boundary.

79.4.2 Tamper Region Override (TO) DCF client

This DCF client holds override information for each TDR. *Figure 1608* shows the diagram for this register.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the override of a TDR permanent.

DCF Client Address: 0x0001 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1608. Tamper Region Override (TO) DCF client

Table 1569. TO field descriptions

Field	Description
26:31 TOE[5:0]	TDR Override Enable Each TOE field, where *n* represents the TDRs, holds information of the Override Enable status for each TDR. 0 Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written 1 Diary override. The diary for this TDR can be erased WITHOUT writing to the diary

79.4.3 One Time Programmable Enable (OTPEN0) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK0 register of the flash memory. *Figure 1609* depicts this client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to OTP irreversible.

DCF Client Address: 0x0008 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	LOWOTPEN											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	MIDOTPEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1609. One Time Programmable Enable (OTPEN0) DCF client

Table 1570. OTPEN0 field descriptions

Field	Description
4:15 LOWOTPEN	Low Block OTP Enable This field affects flash memory blocks in Low address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP
16:31 MIDOTPEN	Mid Block OTP Enable This field affects flash memory blocks in Mid address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

79.4.4 One Time Programmable Enable (OTPEN1) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK1 register of the flash memory. *Figure 1610* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to OTP irreversible.

DCF client Address: 0x0009 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	HIGHOTPEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1610. One Time Programmable Enable (OTPEN1) DCF client

Table 1571. OTPEN1 field descriptions

Field	Description
16:31 HIGHOTPEN	High Block OTP Enable This field affects flash memory blocks in High address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. (Refer to the embedded memories chapter). 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

79.4.5 One Time Programmable Enable (OTPEN2) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK2 register of the flash memory. *Figure 1611* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to OTP irreversible.

DCF Client Address: 0x000A Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	256KBOTPEN[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	256KBOTPEN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1611. One Time Programmable Enable (OTPEN2) DCF client

Table 1572. OTPEN2 field descriptions

Field	Description
0–31 256KBOTPEN	256 KB Block OTP Enable (256KBOTPEN[31:0]) This field affects flash memory blocks in 256KB address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

79.4.6 One Time Programmable Enable (OTPEN3) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF Client mirrors the LOCK3 register of the flash memory. *Figure 1612* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to OTP irreversible.

DCF client Address: 0x000B Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	256KBOTPEN[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1612. One Time Programmable Enable (OTPEN3) DCF client

Table 1573. OTPEN3 field descriptions

Field	Description
16–31 256KBOTPEN	256 KB Block OTP Enable (256KBOTPEN[47:32]) This field affects flash memory blocks in the upper range of 256 KB address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

79.4.7 Tamper Region Assignment DCF client (TDR*n*_LOCK0)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF client however holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. Therefore, the diagram of this register mirrors the LOCK0 register in the flash memory. *Figure 1613* depicts the TDR*n*_LOCK0 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 0) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	LOWTDRAGN											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	MIDTDRAGN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1613. Tamper Region Assignment DCF client (TDR*n*_LOCK0)

Table 1574. TDRn_LOCK0 field descriptions

Field	Description
4:15 LOWTDRAGN	Low Block Tamper Region Assignment This field affects flash memory blocks in Low address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*
16:31 MIDTDRAGN	Mid Block Tamper Region Assignment This field affects flash memory blocks in Mid address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

79.4.8 Tamper Region Assignment DCF client (TDR*n* _LOCK1)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF client however holds the assignment information for only those blocks specified in the LOCK1 register in the flash memory. Therefore, the diagram of this DCF client mirrors the LOCK1 register in the flash memory. *Figure 1614* depicts the TDR*n*_LOCK1 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

DCF client Address: 0x14 + ((n * 4) + 1) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	HIGHTDRAGN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1614. Tamper Region Assignment DCF client (TDR*n* _LOCK1)

Table 1575. TDR*n* _LOCK1 field descriptions

Field	Description
16:31 HIGHTDRAGN	High Block Tamper Region Assignment This field affects flash memory blocks in High address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

79.4.9 Tamper Region Assignment DCF client (TDR*n*_LOCK2)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF Client however holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. Therefore, the diagram of this DCF Client mirrors the LOCK2 register in the flash memory. *Figure 1615* depicts the TDR*n*_LOCK2 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 2) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	256KBTDRAGN[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	256KBTDRAGN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1615. Tamper Region Assignment DCF client (TDR*n*_LOCK2)

Table 1576. TDR*n*_LOCK2 field descriptions

Field	Description
0:31 256KBTDRAGN	256 KB Block Tamper Region Assignment (256KBTDRAGN[31:0]) This field affects flash memory blocks in 256 KB address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

79.4.10 Tamper Region Assignment DCF client (TDR*n*_LOCK3)

This DCF Clients holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF Clients however holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. Therefore, the diagram of this DCF Clients mirrors the LOCK3 register in the flash memory. *Figure 1616* depicts the TDRn_LOCK3 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 3) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	256KBTDRAGN[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1616. Tamper Region Assignment DCF client (TDR*n*_LOCK3)

Table 1577. TDR*n*_LOCK3 field descriptions

Field	Description
16:31 256KBTDRAGN	256 KB Block Tamper Region Assignment (256KBTDRAGN[47:32]) This field affects flash memory blocks in 256 KB address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

79.5 Memory Map and Registers

Note: The Tamper Detection Module's memory-mapped registers are read-only. Any attempted write generates a transfer error.

Table 1578. TDM memory map

Address offset (hex)	Register name	Location
0x00	TDR Status Register (TDM_TDRSR)	*Section 79.5.1*
0x04	Last Flash Programmed Address Register (TDM_LFPAR)	*Section 79.5.2*
0x08	Diary Base Address (TDM_DBA)	*Section 79.5.3*
0x10–0x24	Software Tamper Override Key Region (TDM_STO_KEY0–TDM_STO_KEY5)	*Section 79.5.4*

79.5.1 TDR Status Register (TDM_TDRSR)

This register contains the current status, such as locked or unlocked, of each tamper region, such as TDR.

- If a tamper region is locked, the blocks assigned to that tamper region cannot be erased.
- If a region is unlocked, the blocks within the tamper region can be erased.

The register contains a status bit for each TDR. Status is determined as follows:

1. The default status for each TDR is ‘1’, which indicates that the erase operation is disabled for all flash blocks associated with the TDR.
2. The status bit for a TDR is set to ‘0’ (indicating that the erase operation for all flash blocks associated with the TDR is enabled) if either a successful programming operation to the TDR diary region has been performed or the TDR Override bit for that TDR is set.

Note: TDRSR is a read-only register and writes have no effect. Write operations result in a transfer error.

If the reset value for this register is determined by a DCF record, it will be noted in the chip-specific information at the beginning of this chapter. Otherwise, the reset value is as shown.

Offset: 0x00 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDRSR5	TDRSR4	TDRSR3	TDRSR2	TDRSR1	TDRSR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 1617. TDR Status and Diary Override Register (TDM_TDRSR)

Table 1579. TDM_TDRSR field descriptions

Field	Description
26 TDRSR5	Represents the status of TDR 5 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
27 TDRSR4	Represents the status of TDR 4 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
28 TDRSR3	Represents the status of TDR 3 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

Table 1579. TDM_TDRSR field descriptions (continued)

Field	Description
29 TDRSR2	Represents the status of TDR 2 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
30 TDRSR1	Represents the status of TDR 1 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
31 TDRSR0	Represents the status of TDR 0 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

79.5.2 Last Flash Programmed Address Register (TDM_LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Writes to this register are ignored and will generate a transfer error.

Offset: 0x04 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LFPAR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LFPAR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1618. Last Flash Programmed Address Register (TDM_LFPAR)

Table 1580. TDM_LFPAR field descriptions

Field	Description
8:31 LFPAR	Last Flash Programmed Address These bits represent the address of the last successful programming operation.

79.5.3 Diary Base Address (TDM_DBA)

This register holds base address information of the diary. Any write to this register is ignored and generates a transfer error.

Offset: 0x08 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBA															
W																
Reset	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DBA															
W																
Reset	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)

Figure 1619. Diary Base Address Register (TDM_DBA)

1. Reset value depends on the Diary Base Address (DBA) DCF client.

Table 1581. TDM_DBA field descriptions

Field	Description
0:31 DBA	Diary Base Address These bits represent the address of the diary location in the flash. **Note:** The diary base address is established via the DBA DCF record. The reset value of this register is determined by that record.

79.5.4 Software Tamper Override Key Region (TDM_STO_KEY*n*)

The STO_KEYn register holds the service key for overriding the Tamper Detect Region *n*, where *n* denotes the Tamper Detect Region number.

Caution: The STO_KEY*n* registers can only be written once.
The associated DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

Offset: 0x10 + *n* * 0x4 (*n* = 0 to 5) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STO_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	STO_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1620. Software Tamper Override Key Region (TDM_STO_KEY*n*)

Table 1582. TDM_STO_KEY*n* field descriptions

Field	Description
0:31 STO_KEY	Software Tamper Override Key Diary *n* This field holds the service key value for overriding Tamper Detect Region n. By loading STO_KEYn with the correct signature of 0x55AA5A5A, Tamper Detect Region n is overridden, and the flash blocks assigned to Tamper Detect Region n are open for erase. This field has no effect when STO_DISn field of STO_DIS DCF client is set. **Note:** The STO_KEYn registers can only be written once. **Note:** Refer to the section *Section 52.3.6: TDM Diary Operation* for programming details.

79.6 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

79.6.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the Dairy is defined by a DCF Record .

The diary is divided into 6 parts where each part consists of 256 × 128-bit records, to record until 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the Diary should reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4 × 32-bit internal hardware registers that are initialized by DCF records that define the tamper region . These registers mirror exactly the LOCK registers bits from the flash module.

Each TDRn_LOCK*m* register is loaded at boot time by the SSCM , which defines if the associated flash block is Locked for Erase in a particular TDR:

- ’1’ means the flash block is locked for erase
- ’0’ means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR.As the diary should be OTP, successive writes are allowed on unprogrammed double words only. This ensure that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double word program operation, the flash module indicates to the TDM the address of the programmed double word. TDM determines whether the programmed data was written to the correct TDR Diary Location and then unlocks the blocks for erase operation for the corresponding TDR. TDM ensures that this erase operation on unlocked blocks is allowed until the next program operation on any block, other than Diary, is performed with program completion interrupt enabled or till next hard reset.

Note: All blocks associated to a TDR where an erase operation is allowed can be erased all together.

TDM has no control of how much data a diary has loaded. It is a responsibility for the user to verify if the diary is full. There is also no restriction to what type of data can be programmed in the diary. If diary is detected to be full, TDM provides a means to disable the tamper detect mechanism so that erase operations can be performed to a TDR without any diary entry. In this case, the user is required to program the Tamper Region Override Register for the corresponding TDR.

The data programmed in the diary location is flexible, and may include count information or other important customer information.

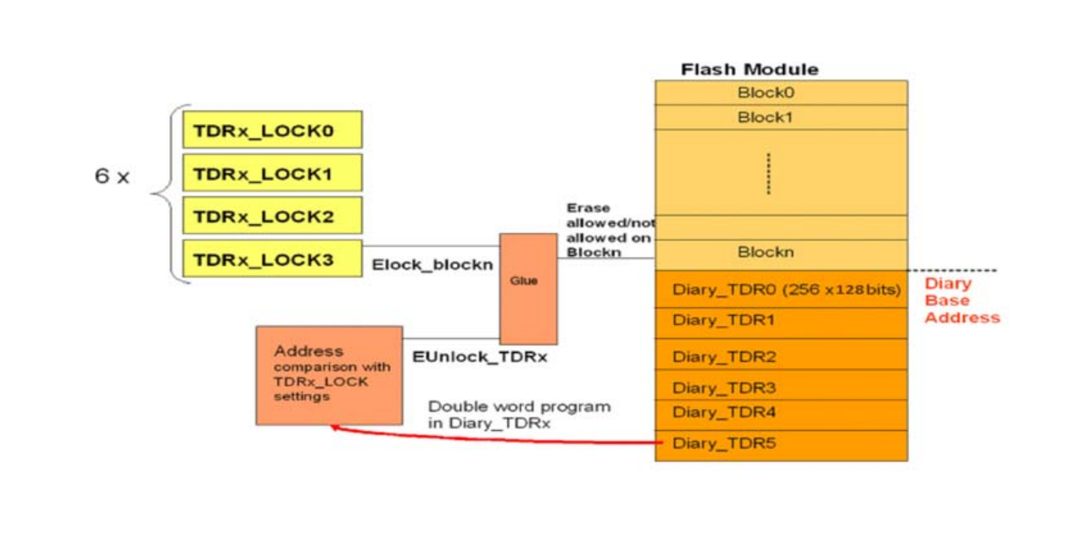
Software has to check whether the diary is full. Once the diary area for a TDR is full, the software has to program a DCF record for overriding blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry

The setting from “override tamper region lock” DCF record (refer to *Section 79.6.3: Diary*) has priority over the settings in the TDRn_LOCK0-3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDR*n*_LOCK*m* bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, the settings from PASS_LOCK*x*_PG*n* registers have priority over TDRn_LOCK*m* settings. For example, if a block is password-protected for write operations, the TDR lock setting is ignored.

A schematic representation of the diary is depicted in *Figure 1621*.

Figure 1621. Diary implementation



Note: The settings in the Tamper Region Override Register have priority over the TDRn_LOCKm Registers. The SSCM module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via TDM, but the same blocks are programmed to be locked for erase operation in SSCM, the blocks end up being locked for erase operation.

79.6.2 Assigning blocks to Tamper Detect Regions (TDR)

Each TDR is defined by a group of 4 × 32-bit DCF records and clients, with each containing bits that map to specific blocks in a flash region. These mappings mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as TDRn_LOCKm where n is the number of a tamper region, that is, 0-5 and m varies from 0-3 as flash contains four Lock registers .

In these DCF records:

- ’1’ means that the corresponding flash block is assigned to this TDR
- ’0’ means that the corresponding flash block is not assigned to this TDR (Default)

DCF clients are not visible to software. They cannot be read nor written. They can only be initialized by DCF record within the UTEST flash block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that correspond to these clients.

Note: No flash block should be assigned to more than one TDR. Each block should correspond to a single TDR. For example, if a single block is assigned to TDR0 and TDR1, then even if the Diary is programmed for both TDR0 and TDR1, this flash block will never be unlocked for erase operation.

79.6.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 4 KB, and the minimum size of any programming operation depends on the Flash sector, which is 8 bytes for Data sector and 16 bytes for the remaining sector to honor OTP restrictions. Therefore, each diary can hold a maximum of two-hundred and fifty-six 128-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core.

Note: To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once during reset and the default bit state is 1.

The DCF clients cannot be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

79.6.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records are defined by the customer. The only hardware constraints are that each diary has a maximum size of 4 KB and the minimum size of any programming operation depends on the Flash sector, which is 8 bytes for Data sector and 16 bytes for the remaining sector.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once during reset and the default bit state is ‘0’.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 4 KB section of the diary. With 6 TDRs, 24 KB of diary space is required.

The diary base address must be at a 16 KB boundary. Therefore the least significant 14 bits of the base address must all be ‘0’.

The Base address of the diary is represented by:

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xxxx_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b000_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b001_0000_0000_0000
- TDR2 = Diary_Base_Address + 0b010_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b011_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b100_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b101_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b110_0000_0000_0000

79.6.5 Diary Base Address Verification

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. The default state of these bits is 1, which is defined as “TDR Erase Blocked”. Refer to *Section 79.5.1: TDR Status Register (TDM_TDRSR)* for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programmed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning “TDR Erase Enabled”.

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 6 status bits can be read through the TDRSR. Refer to *Section 79.5.1: TDR Status Register (TDM_TDRSR)*. The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. Refer to *Section 79.5.2: Last Flash Programmed Address Register (TDM_LFPAR)*.

79.6.5.1 Diary management

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location that can be written. After a maximum of 256 records are programmed to the diary, a TDR diary is full. Full means that at least 1 bit has been programmed to ‘0’ in each record within the 4 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits, one for each TDR. Refer to the device configuration information at the beginning of this chapter for TDM DCF client details.

The Diary Override bits are by default set to ‘0’; which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to ‘1’. Writes to ‘0’ are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to ‘1’. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

79.6.6 One Time Programmable (OTP)

79.6.6.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and that only 64-bit double words that are already erased (that is, flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block will become OTP after the next reset.

80 Wakeup unit (WKPU)

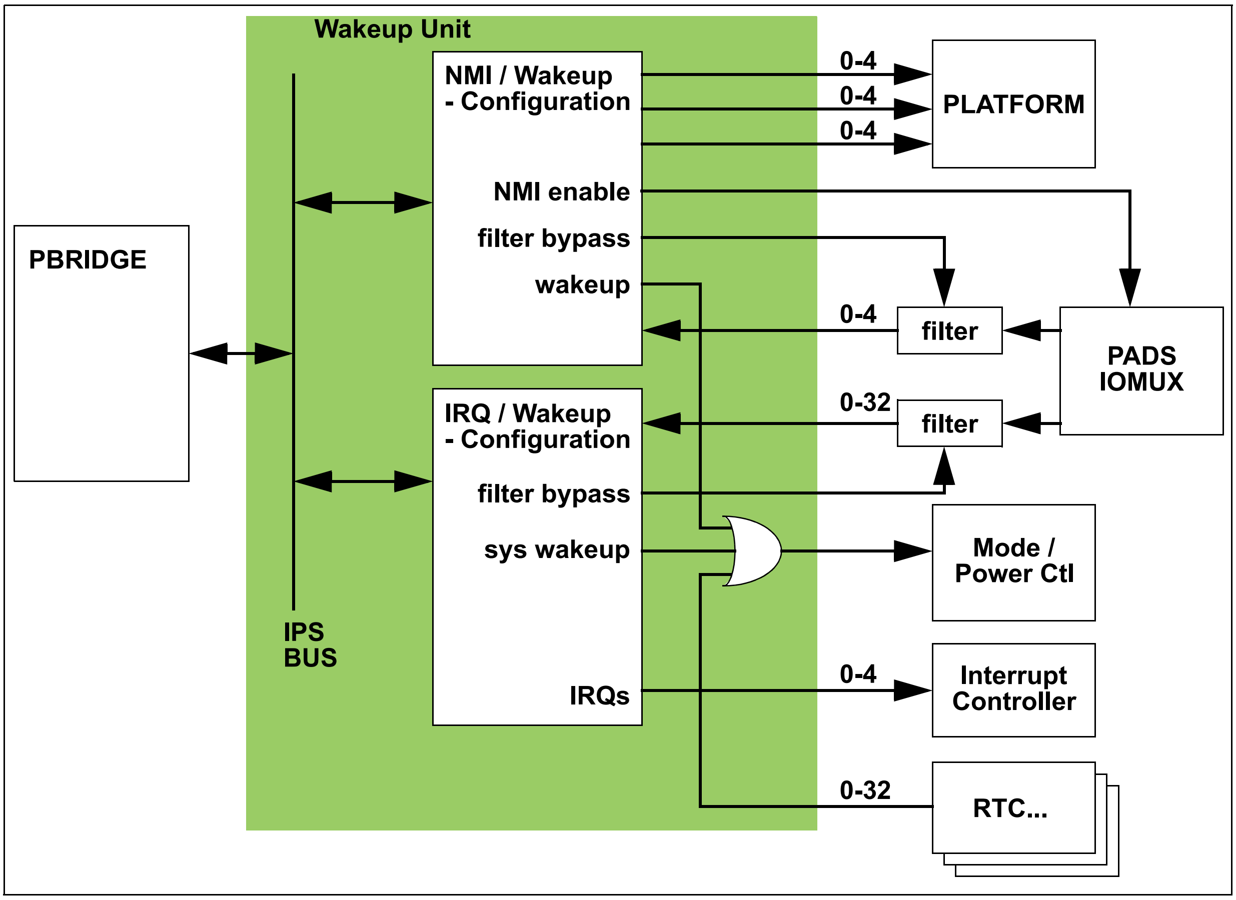
80.1 Introduction

80.1.1 Overview

The Wakeup unit supports up to 32 external sources that can generate CPU exceptions or wakeup events and up to 3 external sources that can cause non-maskable interrupt requests or wakeup events. In addition, it combines its wakeup events with those generated from up to 32 other wakeup sources to supply a single wakeup to the system. This NMI external source can also generate a functional reset event to the MC_RGM. *Figure 1622* is a block diagram of the Wakeup unit and its interfaces to other system components.

Note: For the chip-specific implementation details of this module refer to the Device Configuration chapter.

Figure 1622. Wakeup unit block diagram



80.1.2 Features

The Wakeup unit supports these distinctive features:

- Non-maskable interrupt support with:
  – Up to 4 NMI sources
  – Up to 4 analog glitch filters
  – Independent CPU exceptions destination: non-maskable interrupt, critical interrupt, or machine check request
  – Edge detection
  – Configurable system wakeup triggering from all NMI sources
- External wakeup/interrupt support with:
  – Up to 4 system interrupt vectors for up to 32 interrupt sources
  – Up to 32 analog glitch filters
  – Independent interrupt mask
  – Edge detection
  – Configurable system wakeup triggering from all interrupt sources
  – Configurable pull functionality
- On-chip wakeup support with:
  – Up to 32 wakeup sources
  – Wakeup status mapped to same register as external wakeup/interrupt status

Note: For the chip-specific feature details of this module refer to the Device Configuration chapter.

80.2 External signal description

The Wakeup unit has 0 to 32 signal inputs that can be used as external interrupt sources in normal run mode or as system wakeup sources in certain power down modes.

The module has 0 to 4 signal inputs that can be used as non-maskable interrupt sources in normal run mode or as system wakeup sources in certain power down modes.

80.3 Memory map and register description

This section provides a detailed description of all registers accessible in the WKPU module.

80.3.1 Memory map

Table 1583 gives an overview on the WKPU registers implemented.

Table 1583. WKPU memory map

Address offset	Registers	Supported access sizes	Location
0x0000	NMI Status Flag Register (NSR)	32/16/8	*Section 80.3.2.1*
0x0004–0x0007	Reserved		
0x0008	NMI Configuration Register (NCR)	32/16/8	*Section 80.3.2.2*

Table 1583. WKPU memory map (continued)

Address offset	Registers	Supported access sizes	Location
0x000C–0x0013	Reserved		
0x0014	Wakeup/Interrupt Status Flag Register (WISR)	32	*Section 80.3.2.3*
0x0018	Interrupt Request Enable Register (IRER)	32	*Section 80.3.2.4*
0x001C	Wakeup Request Enable Register (WRER)	32	*Section 80.3.2.5*
0x0020–0x0027	Reserved		
0x0028	Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)	32	*Section 80.3.2.6*
0x002C	Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)	32	*Section 80.3.2.7*
0x0030	Wakeup/Interrupt Filter Enable Register (WIFER)	32	*Section 80.3.2.8*
0x0034	Wakeup/Interrupt Pullup Enable Register (WIPUER)	32	*Section 80.3.2.9*
0x0038–0x3FFF	Reserved		

Note: Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.

80.3.2 Register descriptions

This section describes in address order all the Wakeup unit registers.

80.3.2.1 NMI Status Flag Register (NSR)

This register holds the non-maskable interrupt status flags.

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NIF2	NOVF2	0	0	0	0	0	0	RIF	ROVF	0	0	0	0	0	0
W																	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1623. NMI Status Flag Register (NSR)

Table 1584. NSR field descriptions

Field	Description
16 NIF2	NMI Status Flag 2 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE2 or NFEE2 set), NIF2 causes an exception to the associated CPU, as configured within the NDSS bit of the NCR register. In addition this event can also be configured to trigger a system wakeup. 1 An event as defined by NREE2 and NFEE2 has occurred. 0 No event has occurred on the pad.
17 NOVF2	NMI Overrun Status Flag 2 This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF2 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE2 or NFEE2 set), NOVF2 causes an interrupt request. 1 An overrun has occurred on NMI input 2. 0 No overrun has occurred on NMI input 2.
24 RIF	ESR1 External Reset status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (that is if RREE or RFEE is set), RIF status bit is set and will cause a functional reset request to RGM if RDSS[1:0] of NCR is ‘00’. This bit is automatically cleared if the F_WAKEUP reset request is configured in the MC_RGM to generate a long or short “functional” reset. 1 An event as defined by RREE and RFEE has occurred. 0 No event has occurred on the pad.
25 ROVF	ESR1 External Reset overrun status flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current RIF value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (RREE or RFEE set), ROVF can cause a reset request to MC_RGM if RDSS[1:0] of NCR is ‘00’. This bit is automatically cleared if the F_WAKEUP reset request is configured in the MC_RGM to generate a long or short “functional” reset. 1 An overrun has occurred on NMI input. 0 No overrun has occurred on NMI input.

Note: Refer to the Device Configuration chapter to see how the NMI Status Flag Register (NSR) is used in the device.

80.3.2.2 NMI Configuration Register (NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NLOCK2	NDSS2		NWRE2	0	NREE2	NFEE2	NFE2	RLOCK	RDSS		RWRE	0	RREE	RFEE	RFE(1)
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 1624. NMI Configuration Register (NCR)

1. Writing this bit has no functional impact.

Table 1585. NCR field descriptions

Field	Description
16 NLOCK2	NMI Configuration Lock Register 2 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.
17:18 NDSS2	NMI Destination Source Select 2 This value is reset on system reset. 00 Non-maskable interrupt 01 Critical interrupt 10 Machine check request 11 Reset request
19 NWRE2	NMI Wakeup Request Enable 2 1 NMI2 wakeup is enabled. A set NIF2 bit or set NOVF2 bit causes a system wakeup request. 0 NMI2 wakeup is disabled. System wakeup requests from the corresponding NIF2 bit are disabled.
21 NREE2	NMI Rising-edge Events Enable 2 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
22 NFEE2	NMI Falling-edge Events Enable 2 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
23 NFE2	NMI Filter Enable 2 Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.
24 RLOCK	ESR1 External Reset Configuration Lock Register Writing a 1 to this bit locks the configuration for the reset until it is unlocked by a system reset. Writing a 0 has no effect.
25:26 RDSS	ESR1 External Reset Destination Source Select This value is reset on system reset. 00 Reset request to RGM 01 No reaction 10 No reaction 11 No reaction
27 RWRE	ESR1 External Reset Wakeup Request Enable 1 A set RIF bit or set ROVF bit causes a system wakeup request. 0 System wakeup requests from the corresponding RIF bit are disabled. **Note:** To wake up the system from standby by this way, ESR1 option has to be disable in MC_RGM. To disable the MC_RGM ESR1 function, the bit 30 of RGM_FERD and RGM_FEAR registers have to be set.
29 RREE	ESR1 External Reset Rising-edge Events Enable 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.

Table 1585. NCR field descriptions (continued)

Field	Description
30 RFEE	ESR1 External Reset Falling-edge Events Enable 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
31 RFE	ESR1 External Reset Filter Enable Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.

Note: Refer to the Device configuration chapter to see how the NMI Configuration Register (NCR) is used in the device.

Note: Writing a '0' to both NREEx and NFEEx disables the NMI functionality completely (no system wakeup or interrupt will be generated on any pad activity).

80.3.2.3 Wakeup/Interrupt Status Flag Register (WISR)

This register holds the wakeup/interrupt flags.

Offset: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIF31	EIF30	EIF29	EIF28	EIF27	EIF26	EIF25	EIF24	EIF23	EIF22	EIF21	EIF20	EIF19	EIF18	EIF17	EIF16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1625. Wakeup/Interrupt Status Flag Register (WISR)

Table 1586. WISR field descriptions

Field	Description
0 EIF31	External Wakeup/Interrupt Status Flag 31 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER31), EIF31 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
1 EIF30	External Wakeup/Interrupt Status Flag 30 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER30), EIF30 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
2 EIF29	External Wakeup/Interrupt Status Flag 29 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER29), EIF29 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Table 1586. WISR field descriptions (continued)

Field	Description
3 EIF28	External Wakeup/Interrupt Status Flag 28 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER28), EIF28 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
4 EIF27	External Wakeup/Interrupt Status Flag 27 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER27), EIF27 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
5 EIF26	External Wakeup/Interrupt Status Flag 26 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER26), EIF26 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
6 EIF25	External Wakeup/Interrupt Status Flag 25 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER25), EIF25 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
7 EIF24	External Wakeup/Interrupt Status Flag 24 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER24), EIF24 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
8 EIF23	External Wakeup/Interrupt Status Flag 23 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER23), EIF23 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
9 EIF22	External Wakeup/Interrupt Status Flag 22 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER22), EIF22 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
10 EIF21	External Wakeup/Interrupt Status Flag 21 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER21), EIF21 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
11 EIF20	External Wakeup/Interrupt Status Flag 20 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER20), EIF20 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Table 1586. WISR field descriptions (continued)

Field	Description
12 EIF19	External Wakeup/Interrupt Status Flag 19 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER19), EIF19 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
13 EIF18	External Wakeup/Interrupt Status Flag 18 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER18), EIF18 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
14 EIF17	External Wakeup/Interrupt Status Flag 17 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER17), EIF17 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
15 EIF16	External Wakeup/Interrupt Status Flag 16 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER16), EIF16 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
16 EIF15	External Wakeup/Interrupt Status Flag 15 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER15), EIF15 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
17 EIF14	External Wakeup/Interrupt Status Flag 14 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER14), EIF14 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
18 EIF13	External Wakeup/Interrupt Status Flag 13 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER13), EIF13 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
19 EIF12	External Wakeup/Interrupt Status Flag 12 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER12), EIF12 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
20 EIF11	External Wakeup/Interrupt Status Flag 11 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER11), EIF11 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Table 1586. WISR field descriptions (continued)

Field	Description
21 EIF10	External Wakeup/Interrupt Status Flag 10 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER10), EIF10 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
22 EIF9	External Wakeup/Interrupt Status Flag 9 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER9), EIF9 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
23 EIF8	External Wakeup/Interrupt Status Flag 8 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER8), EIF8 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
24 EIF7	External Wakeup/Interrupt Status Flag 7 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER7), EIF7 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
25 EIF6	External Wakeup/Interrupt Status Flag 6 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER6), EIF6 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
26 EIF5	External Wakeup/Interrupt Status Flag 5 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER5), EIF5 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
27 EIF4	External Wakeup/Interrupt Status Flag 4 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER4), EIF4 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
28 EIF3	External Wakeup/Interrupt Status Flag 3 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER3), EIF3 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
29 EIF2	External Wakeup/Interrupt Status Flag 2 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER2), EIF2 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Table 1586. WISR field descriptions (continued)

Field	Description
30 EIF1	External Wakeup/Interrupt Status Flag 1 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER1), EIF1 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
31 EIF0	External Wakeup/Interrupt Status Flag 0 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER0), EIF0 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Note: Status bits associated with on-chip wakeup sources are located to the left of the external wakeup/interrupt status bits and are read only. The wakeup for these sources must be configured and cleared at the on-chip wakeup source. Also, the configuration registers for the external interrupts/wakeups do not have corresponding bits.

80.3.2.4 Interrupt Request Enable Register (IRER)

This register is used to enable the interrupt messaging from the wakeup/interrupt pads to the interrupt controller.

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EIRE31	EIRE30	EIRE29	EIRE28	EIRE27	EIRE26	EIRE25	EIRE24	EIRE23	EIRE22	EIRE21	EIRE20	EIRE19	EIRE18	EIRE17	EIRE16	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1626. Interrupt Request Enable Register (IRER)

Table 1587. IRER field descriptions

Field	Description
0 EIRE31	External Interrupt Request Enable 31 1 A set EIF31 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF31 bit are disabled.
1 EIRE30	External Interrupt Request Enable 30 1 A set EIF30 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF30 bit are disabled.
2 EIRE29	External Interrupt Request Enable 29 1 A set EIF29 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF29 bit are disabled.
3 EIRE28	External Interrupt Request Enable 28 1 A set EIF28bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF28 bit are disabled.

Table 1587. IRER field descriptions (continued)

Field	Description
4 EIRE27	External Interrupt Request Enable 27 1 A set EIF27 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF27 bit are disabled.
5 EIRE26	External Interrupt Request Enable 26 1 A set EIF26 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF26 bit are disabled.
6 EIRE25	External Interrupt Request Enable 25 1 A set EIF25 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF25 bit are disabled.
7 EIRE24	External Interrupt Request Enable 24 1 A set EIF24 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF24 bit are disabled.
8 EIRE23	External Interrupt Request Enable 23 1 A set EIF23 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF23 bit are disabled.
9 EIRE22	External Interrupt Request Enable 22 1 A set EIF22bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF22 bit are disabled.
10 EIRE21	External Interrupt Request Enable 21 1 A set EIF21 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF21 bit are disabled.
11 EIRE20	External Interrupt Request Enable 20 1 A set EIF20 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF20 bit are disabled.
12 EIRE19	External Interrupt Request Enable 19 1 A set EIF19 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF19 bit are disabled.
13 EIRE18	External Interrupt Request Enable 18 1 A set EIF18 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF18 bit are disabled.
14 EIRE17	External Interrupt Request Enable 17 1 A set EIF17bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF17 bit are disabled.
15 EIRE16	External Interrupt Request Enable 16 1 A set EIF16 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF16 bit are disabled.
16 EIRE15	External Interrupt Request Enable 15 1 A set EIF15 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF15 bit are disabled.
17 EIRE14	External Interrupt Request Enable 14 1 A set EIF14 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF14 bit are disabled.

Table 1587. IRER field descriptions (continued)

Field	Description
18 EIRE13	External Interrupt Request Enable 13 1 A set EIF13 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF13 bit are disabled.
19 EIRE12	External Interrupt Request Enable 12 1 A set EIF12 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF12 bit are disabled.
20 EIRE11	External Interrupt Request Enable 11 1 A set EIF11 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF11 bit are disabled.
21 EIRE10	External Interrupt Request Enable 10 1 A set EIF10 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF10 bit are disabled.
22 EIRE9	External Interrupt Request Enable 9 1 A set EIF9 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF9 bit are disabled.
23 EIRE8	External Interrupt Request Enable 8 1 A set EIF8 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF8 bit are disabled.
24 EIRE7	External Interrupt Request Enable 7 1 A set EIF7 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF7 bit are disabled.
25 EIRE6	External Interrupt Request Enable 6 1 A set EIF6 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF6 bit are disabled.
26 EIRE5	External Interrupt Request Enable 5 1 A set EIF5 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF5 bit are disabled.
27 EIRE4	External Interrupt Request Enable 4 1 A set EIF4 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF4 bit are disabled.
28 EIRE3	External Interrupt Request Enable 3 1 A set EIF3 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF3 bit are disabled.
29 EIRE2	External Interrupt Request Enable 2 1 A set EIF2 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF2 bit are disabled.
30 EIRE1	External Interrupt Request Enable 1 1 A set EIF1 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF1 bit are disabled.
31 EIRE0	External Interrupt Request Enable 0 1 A set EIF0 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF0 bit are disabled.

80.3.2.5 Wakeup Request Enable Register (WRER)

This register is used to enable the system wakeup messaging from the wakeup/interrupt pads to the mode entry and power control modules.

Note: Any wakeup pad must be properly driven either externally or internally (enabling weak pull-up/pull-down) before enabling its corresponding WREx bit.

Offset: 0x001C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRE31	WRE30	WRE29	WRE28	WRE27	WRE26	WRE25	WRE24	WRE23	WRE22	WRE21	WRE20	WRE19	WRE18	WRE17	WRE16	WRE15	WRE14	WRE13	WRE12	WRE11	WRE10	WRE9	WRE8	WRE7	WRE6	WRE5	WRE4	WRE3	WRE2	WRE1	WRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1627. Wakeup Request Enable Register (WRER)

Table 1588. WRER field descriptions

Field	Description
0 WRE31	External Wakeup Request Enable 31 1 A set EIF31 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF31 bit are disabled.
1 WRE30	External Wakeup Request Enable 30 1 A set EIF30 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF30 bit are disabled.
2 WRE29	External Wakeup Request Enable 29 1 A set EIF29 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF29 bit are disabled.
3 WRE28	External Wakeup Request Enable 28 1 A set EIF28 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF28 bit are disabled.
4 WRE27	External Wakeup Request Enable 27 1 A set EIF27 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF27 bit are disabled.
5 WRE26	External Wakeup request.Enable 26 1 A set EIF26 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF26 bit are disabled.
6 WRE25	External Wakeup request.Enable 25 1 A set EIF25 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF25 bit are disabled.
7 WRE24	External Wakeup request.Enable 24 1 A set EIF24 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF24 bit are disabled.
8 WRE23	External Wakeup request.Enable 23 1 A set EIF23 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF23 bit are disabled

Table 1588. WRER field descriptions (continued)

Field	Description
9 WRE22	External Wakeup Request Enable 22 1 A set EIF22 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF22 bit are disabled.
10 WRE21	External Wakeup Request Enable 21 1 A set EIF21 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF21 bit are disabled.
11 WRE20	External Wakeup Request Enable 20 1 A set EIF20 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF20 bit are disabled.
12 WRE19	External Wakeup Request Enable 19 1 A set EIF19 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF19 bit are disabled.
13 WRE18	External Wakeup Request Enable 18 1 A set EIF18 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF18 bit are disabled.
14 WRE17	External Wakeup request.Enable 17 1 A set EIF17 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF17 bit are disabled.
15 WRE16	External Wakeup request.Enable 16 1 A set EIF16 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF16 bit are disabled.
16 WRE15	External Wakeup request.Enable 15 1 A set EIF15 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF15 bit are disabled.
17 WRE14	External Wakeup request.Enable 14 1 A set EIF14 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF14 bit are disabled.
18 WRE13	External Wakeup request.Enable 13 1 A set EIF13 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF13 bit are disabled.
19 WRE12	External Wakeup request.Enable 12 1 A set EIF12 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF12 bit are disabled.
20 WRE11	External Wakeup request.Enable 11 1 A set EIF11 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF11 bit are disabled.
21 WRE10	External Wakeup request.Enable 10 1 A set EIF10 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF10 bit are disabled.
22 WRE9	External Wakeup request.Enable 9 1 A set EIF9 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF9 bit are disabled.

Table 1588. WRER field descriptions (continued)

Field	Description
23 WRE8	External Wakeup request.Enable 8. 1 A set EIF8 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF8 bit are disabled
24 WRE7	External Wakeup Request Enable 7 1 A set EIF7 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF7 bit are disabled.
25 WRE6	External Wakeup Request Enable 6 1 A set EIF6 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF6 bit are disabled.
26 WRE5	External Wakeup Request Enable 5 1 A set EIF5 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF5 bit are disabled.
27 WRE4	External Wakeup Request Enable 4 1 A set EIF4 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF4 bit are disabled.
28 WRE3	External Wakeup Request Enable 3 1 A set EIF3 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF3 bit are disabled.
29 WRE2	External Wakeup Request Enable 2 1 A set EIF2 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF2 bit are disabled.
30 WRE1	External Wakeup Request Enable 1 1 A set EIF1 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF1 bit are disabled.
31 WRE0	External Wakeup Request Enable 0 1 A set EIF0 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF0 bit are disabled.

80.3.2.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

This register is used to enable rising-edge triggered events on the corresponding wakeup/interrupt pads.

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IREE31	IREE30	IREE29	IREE28	IREE27	IREE26	IREE25	IREE24	IREE23	IREE22	IREE21	IREE20	IREE19	IREE18	IREE17	IREE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IREE15	IREE14	IREE13	IREE12	IREE11	IREE10	IREE9	IREE8	IREE7	IREE6	IREE5	IREE4	IREE3	IREE2	IREE1	IREE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1628. Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

Table 1589. WIREER field descriptions

Field	Description
0 IREE31	External Interrupt Rising-edge Events Enable 31 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
1 IREE30	External Interrupt Rising-edge Events Enable 30 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
2 IREE29	External Interrupt Rising-edge Events Enable 29 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
3 IREE28	External Interrupt Rising-edge Events Enable 28 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
4 IREE27	External Interrupt Rising-edge Events Enable 27 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
5 IREE26	External Interrupt Rising-edge Events Enable 26 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
6 IREE25	External Interrupt Rising-edge Events Enable 25 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
7 IREE24	External Interrupt Rising-edge Events Enable 24 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
8 IREE23	External Interrupt Rising-edge Events Enable 23 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
9 IREE22	External Interrupt Rising-edge Events Enable 22 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
10 IREE21	External Interrupt Rising-edge Events Enable 21 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
11 IREE20	External Interrupt Rising-edge Events Enable 20 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
12 IREE19	External Interrupt Rising-edge Events Enable 19 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
13 IREE18	External Interrupt Rising-edge Events Enable 18 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.

Table 1589. WIREER field descriptions (continued)

Field	Description
14 IREE17	External Interrupt Rising-edge Events Enable 17 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
15 IREE16	External Interrupt Rising-edge Events Enable 16 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
16 IREE15	External Interrupt Rising-edge Events Enable 15 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
17 IREE14	External Interrupt Rising-edge Events Enable 14 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
18 IREE13	External Interrupt Rising-edge Events Enable 13 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
19 IREE12	External Interrupt Rising-edge Events Enable 12 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
20 IREE11	External Interrupt Rising-edge Events Enable 11 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
21 IREE10	External Interrupt Rising-edge Events Enable 10 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
22 IREE9	External Interrupt Rising-edge Events Enable 9 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
23 IREE8	External Interrupt Rising-edge Events Enable 8 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
24 IREE7	External Interrupt Rising-edge Events Enable 7 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
25 IREE6	External Interrupt Rising-edge Events Enable 6 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
26 IREE5	External Interrupt Rising-edge Events Enable 5 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
27 IREE4	External Interrupt Rising-edge Events Enable 4 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.

Table 1589. WIREER field descriptions (continued)

Field	Description
28 IREE3	External Interrupt Rising-edge Events Enable 3 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
29 IREE2	External Interrupt Rising-edge Events Enable 2 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
30 IREE1	External Interrupt Rising-edge Events Enable 1 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
31 IREE0	External Interrupt Rising-edge Events Enable 0 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.

80.3.2.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

This register is used to enable falling-edge triggered events on the corresponding wakeup/interrupt pads.

Offset: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFEE31	IFEE30	IFEE29	IFEE28	IFEE27	IFEE26	IFEE25	IFEE24	IFEE23	IFEE22	IFEE21	IFEE20	IFEE19	IFEE18	IFEE17	IFEE16	IFEE15	IFEE14	IFEE13	IFEE12	IFEE11	IFEE10	IFEE9	IFEE8	IFEE7	IFEE6	IFEE5	IFEE4	IFEE3	IFEE2	IFEE1	IFEE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1629. Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

Table 1590. WIFEER field descriptions

Field	Description
0 IFEE31	External Interrupt Falling-edge Events Enable 31 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
1 IFEE30	External Interrupt Falling-edge Events Enable 30 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
2 IFEE29	External Interrupt Falling-edge Events Enable 29 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
3 IFEE28	External Interrupt Falling-edge Events Enable 28 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
4 IFEE27	External Interrupt Falling-edge Events Enable 27 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.

Table 1590. WIFEER field descriptions (continued)

Field	Description
5 IFEE26	External Interrupt Falling-edge Events Enable 26 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
6 IFEE25	External Interrupt Falling-edge Events Enable 25 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
7 IFEE24	External Interrupt Falling-edge Events Enable 24 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
8 IFEE23	External Interrupt Falling-edge Events Enable 23 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
9 IFEE22	External Interrupt Falling-edge Events Enable 22 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
10 IFEE21	External Interrupt Falling-edge Events Enable 21 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
11 IFEE20	External Interrupt Falling-edge Events Enable 20 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
12 IFEE19	External Interrupt Falling-edge Events Enable 19 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
13 IFEE18	External Interrupt Falling-edge Events Enable 18 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
14 IFEE17	External Interrupt Falling-edge Events Enable 17 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
15 IFEE16	External Interrupt Falling-edge Events Enable 16 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
16 IFEE15	External Interrupt Falling-edge Events Enable 15 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
17 IFEE14	External Interrupt Falling-edge Events Enable 14 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
18 IFEE13	External Interrupt Falling-edge Events Enable 13 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.

Table 1590. WIFEER field descriptions (continued)

Field	Description
19 IFEE12	External Interrupt Falling-edge Events Enable 12 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
20 IFEE11	External Interrupt Falling-edge Events Enable 11 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
21 IFEE10	External Interrupt Falling-edge Events Enable 10 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
22 IFEE9	External Interrupt Falling-edge Events Enable 9 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
23 IFEE8	External Interrupt Falling-edge Events Enable 8 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
24 IFEE7	External Interrupt Falling-edge Events Enable 7 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
25 IFEE6	External Interrupt Falling-edge Events Enable 6 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
26 IFEE5	External Interrupt Falling-edge Events Enable 5 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
27 IFEE4	External Interrupt Falling-edge Events Enable 4 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
28 IFEE3	External Interrupt Falling-edge Events Enable 3 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
29 IFEE2	External Interrupt Falling-edge Events Enable 2 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
30 IFEE1	External Interrupt Falling-edge Events Enable 1 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
31 IFEE0	External Interrupt Falling-edge Events Enable 0 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.

80.3.2.8 Wakeup/Interrupt Filter Enable Register (WIFER)

This register is used to enable an analog filter on the corresponding interrupt pads to filter out glitches on the inputs. The number of wakeups/interrupts supporting this feature is SoC dependent and can be configured to be between 1 and 32.

Note: Glitch filter must be enabled in case the wake-up sources are used to exit from Standby.

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFE31	IFE30	IFE29	IFE28	IFE27	IFE26	IFE25	IFE24	IFE23	IFE22	IFE21	IFE20	IFE19	IFE18	IFE17	IFE16	IFE15	IFE14	IFE13	IFE12	IFE11	IFE10	IFE9	IFE8	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1630. Wakeup/Interrupt Filter Enable Register (WIFER)

Table 1591. WIFER field descriptions

Field	Description
0 IFE31	External Interrupt Filter Enable 31 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
1 IFE30	External Interrupt Filter Enable 30 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
2 IFE29	External Interrupt Filter Enable 29 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
3 IFE28	External Interrupt Filter Enable 28 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
4 IFE27	External Interrupt Filter Enable 27 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
5 IFE26	External Interrupt Filter Enable 26 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
6 IFE25	External Interrupt Filter Enable 25 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.

Table 1591. WIFER field descriptions (continued)

Field	Description
7 IFE24	External Interrupt Filter Enable 24 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
8 IFE23	External Interrupt Filter Enable 23 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
9 IFE22	External Interrupt Filter Enable 22 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
10 IFE21	External Interrupt Filter Enable 21 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
11 IFE20	External Interrupt Filter Enable 20 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
12 IFE19	External Interrupt Filter Enable 19 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
13 IFE18	External Interrupt Filter Enable 18 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
14 IFE17	External Interrupt Filter Enable 17 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
15 IFE16	External Interrupt Filter Enable 16 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
16 IFE15	External Interrupt Filter Enable 15 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
17 IFE14	External Interrupt Filter Enable 14 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.

Table 1591. WIFER field descriptions (continued)

Field	Description
18 IFE13	External Interrupt Filter Enable 13 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
19 IFE12	External Interrupt Filter Enable 12 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
20 IFE11	External Interrupt Filter Enable 11 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
21 IFE10	External Interrupt Filter Enable 10 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
22 IFE9	External Interrupt Filter Enable 9 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
23 IFE8	External Interrupt Filter Enable 8 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
24 IFE7	External Interrupt Filter Enable 7 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
25 IFE6	External Interrupt Filter Enable 6 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
26 IFE5	External Interrupt Filter Enable 5 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
27 IFE4	External Interrupt Filter Enable 4 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
28 IFE3	External Interrupt Filter Enable 3 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.

Table 1591. WIFER field descriptions (continued)

Field	Description
29 IFE2	External Interrupt Filter Enable 2 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
30 IFE1	External Interrupt Filter Enable 1 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
31 IFE0	External Interrupt Filter Enable 0 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.

80.3.2.9 Wakeup/Interrupt Pull Enable Register (WIPUER)

This register is used to enable a pull on the corresponding interrupt pads. The pull direction (up or down) is selected by SIUL2_MSCR_IO_n.WPDE-WPUE bits.

Note: WIPUER bits related to those bonded LP pads that are used to wake-up the device from STANDBY Mode, should be left at 0 default value.

- *Bits setting has undesired effect in STANDBY mode (extra consumption or anticipated exiting from STANDBY mode)*
- *In running mode, pull-up/down IO configuration shall be done only through SIUL*

Offset: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	IPUE31	IPUE30	IPUE29	IPUE28	IPUE27	IPUE26	IPUE25	IPUE24	IPUE23	IPUE22	IPUE21	IPUE20	IPUE19	IPUE18	IPUE17	IPUE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	IPUE15	IPUE14	IPUE13	IPUE12	IPUE11	IPUE10	IPUE9	IPUE8	IPUE7	IPUE6	IPUE5	IPUE4	IPUE3	IPUE2	IPUE1	IPUE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1631. Wakeup/Interrupt Pullup Enable Register (WIPUER)

Table 1592. WIPUER field descriptions

Field	Description(1)
0 IPUE31	External Interrupt Pull Enable 31 1 Pull is enabled. 0 Pull is disabled.
1 IPUE30	External Interrupt Pull Enable 30 1 Pull is enabled. 0 Pull is disabled.
2 IPUE29	External Interrupt Pull Enable 29 1 Pull is enabled. 0 Pull is disabled.

Table 1592. WIPUER field descriptions (continued)

Field	Description[1]
3 IPUE28	External Interrupt Pull Enable 28 1 Pull is enabled. 0 Pull is disabled.
4 IPUE27	External Interrupt Pull Enable 27 1 Pull is enabled. 0 Pull is disabled.
5 IPUE26	External Interrupt Pull Enable 26 1 Pull is enabled. 0 Pull is disabled.
6 IPUE25	External Interrupt Pull Enable 25 1 Pull is enabled. 0 Pull is disabled.
7 IPUE24	External Interrupt Pull Enable 24 1 Pull is enabled. 0 Pull is disabled.
8 IPUE23	External Interrupt Pull Enable 23 1 Pull is enabled. 0 Pull is disabled.
9 IPUE22	External Interrupt Pull Enable 22 1 Pull is enabled. 0 Pull is disabled.
10 IPUE21	External Interrupt Pull Enable 21 1 Pull is enabled. 0 Pull is disabled.
11 IPUE20	External Interrupt Pull Enable 20 1 Pull is enabled. 0 Pull is disabled.
12 IPUE19	External Interrupt Pull Enable 19 1 Pull is enabled. 0 Pull is disabled.
13 IPUE18	External Interrupt Pull Enable 18 1 Pull is enabled. 0 Pull is disabled.
14 IPUE17	External Interrupt Pull Enable 17 1 Pull is enabled. 0 Pull is disabled.
15 IPUE16	External Interrupt Pull Enable 16 1 Pull is enabled. 0 Pull is disabled.
16 IPUE15	External Interrupt Pull Enable 15 1 Pull is enabled. 0 Pull is disabled.

Table 1592. WIPUER field descriptions (continued)

Field	Description(1)
17 IPUE14	External Interrupt Pull Enable 14 1 Pull is enabled. 0 Pull is disabled.
18 IPUE13	External Interrupt Pull Enable 13 1 Pull is enabled. 0 Pull is disabled.
19 IPUE12	External Interrupt Pull Enable 12 1 Pull is enabled. 0 Pull is disabled.disabled.
20 IPUE11	External Interrupt Pull Enable 11 1 Pull is enabled. 0 Pull is disabled.
21 IPUE10	External Interrupt Pull Enable 10 1 Pull is enabled. 0 Pull is disabled.
22 IPUE9	External Interrupt Pull Enable 9 1 Pull is enabled. 0 Pull is disabled.
23 IPUE8	External Interrupt Pull Enable 8 1 Pull is enabled. 0 Pull is disabled.
24 IPUE7	External Interrupt Pull Enable 7 1 Pull is enabled. 0 Pull is disabled.
25 IPUE6	External Interrupt Pull Enable 6 1 Pull is enabled. 0 Pull is disabled.
26 IPUE5	External Interrupt Pull Enable 5 1 Pull is enabled. 0 Pull is disabled.
27 IPUE4	External Interrupt Pull Enable 4 1 Pull is enabled. 0 Pull is disabled.
28 IPUE3	External Interrupt Pull Enable 3 1 Pull is enabled. 0 Pull is disabled.
29 IPUE2	External Interrupt Pull Enable 2 1 Pull is enabled. 0 Pull is disabled.

Table 1592. WIPUER field descriptions (continued)

Field	Description[(1)]
30 IPUE1	External Interrupt Pull Enable 1 1 Pull is enabled. 0 Pull is disabled.
31 IPUE0	External Interrupt Pull Enable 0 1 Pull is enabled. 0 Pull is disabled.

1. For the actual chip-specific bit description details refer to the Device Configuration chapter.

80.4 Functional description

80.4.1 General

This section provides a complete functional description of the Wakeup unit.

80.4.2 Non-maskable interrupts

The Wakeup unit supports up to 4 non-maskable interrupts which can be allocated to any pad necessary at the SoC level. This allocation is fixed per SoC.

The Wakeup unit supports the generation of 3 types of interrupts per NMI input to the SoC. The Wakeup unit supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

Each NMI passes through a by-passable analog glitch filter.

Note: Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

80.4.2.1 NMI management

Each NMI can be enabled or disabled independently. This can be performed using the single NCR register laid out to contain all configuration bits for a given NMI in a single byte (see *Figure 1624*). A pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

Note: After reset, NREE and NFEE are set to ‘0’, therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once a pad’s NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See *Table 1585* for details.

Each NMI supports a status flag and an overrun flag which are located in the NSR register (see *Figure 1623*). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (it has not yet been cleared).

Note: The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

80.4.3 External wakeups/interrupts

The Wakeup unit supports up to 32 external wakeup/interrupts which can be allocated to any pad necessary at the SoC level. This allocation is fixed per SoC.

The Wakeup unit supports up to 4 interrupt vectors to the interrupt controller of the SoC. Each interrupt vector can support up to the number of external interrupt sources from the device pads with the total across all vectors being equal to the number of external interrupt sources. Each external interrupt sources is assigned to exactly one interrupt vector. The interrupt vector assignment is sequential so that one interrupt vector is for external interrupt sources 0 through N - 1, the next is for N through N + M - 1, and so forth.

The parameter IRQ_VEC defines the number of interrupt vectors (the ipi_int_pins output size).

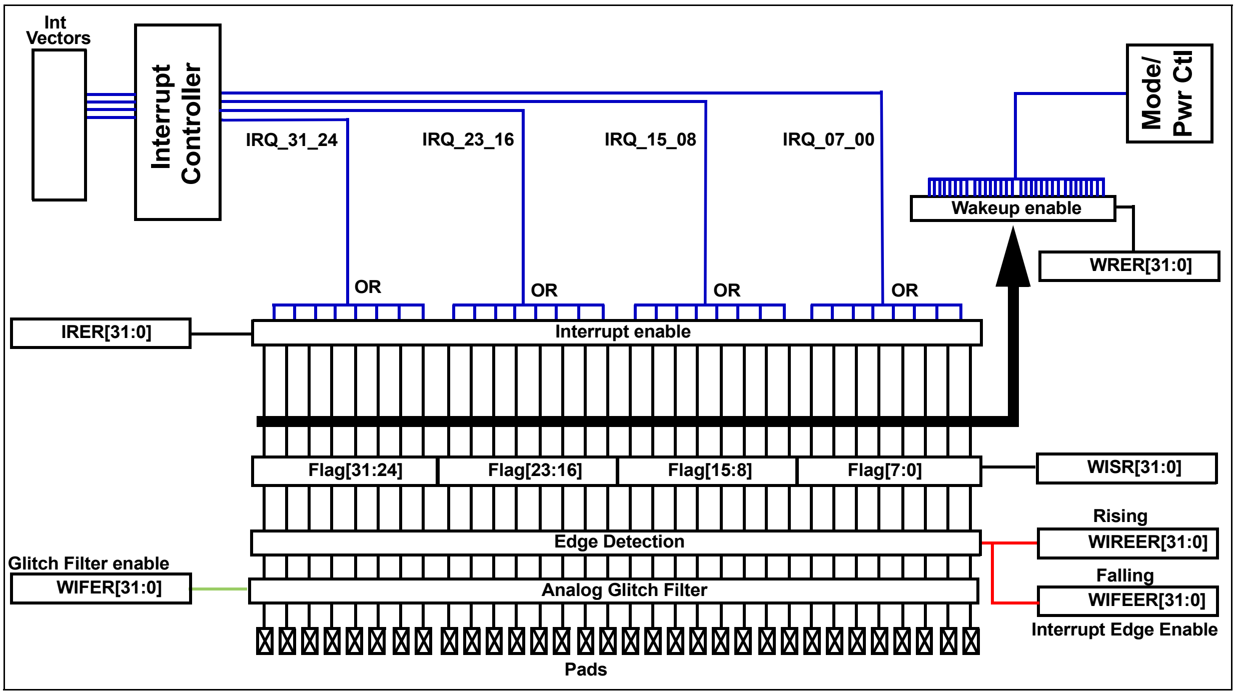
The parameter EIRQ_PER_VEC defines the number of external interrupt sources assigned to each interrupt vector. It is split up into four 6-bit fields, one for each possible interrupt vector, with the lowest field corresponding to ipi_int_pins[0], the second field to ipi_int_pins[1], and so forth.

For example, for EIRQS = 17, IRQ_VEC = 3 and EIRQ_PER_VEC = {6'd0, 6'd5, 6'd3, 6'd9}, ipp_ind_int[8:0] are assigned to ipi_int_pins[0], ipp_ind_int[11:9] are assigned to ipi_int_pins[1], and ipp_ind_int[16:12] are assigned to ipi_int_pins[2]. The signals ipp_ind_int[31:17] and ipi_int_pins[3] do not exist.

Note: If EIRQS = 0, the unused ipp_ind_int[0] input has no effect on the sys_wakeup output.

Refer to *Figure 1632* for an overview of the external interrupt implementation for the example of four interrupt vectors with eight external interrupt sources each.

Figure 1632. External interrupt pad diagram



All of the external interrupt pads within a single group have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the SoC.

An SoC specific number of external interrupt lines can have a digital glitch filters applied to them. The supported range is 0 to 32.

Note: Glitch filter control and pad configuration should be done while the external interrupt line is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

80.4.3.1 External interrupt management

Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (*Figure 1626*). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active.

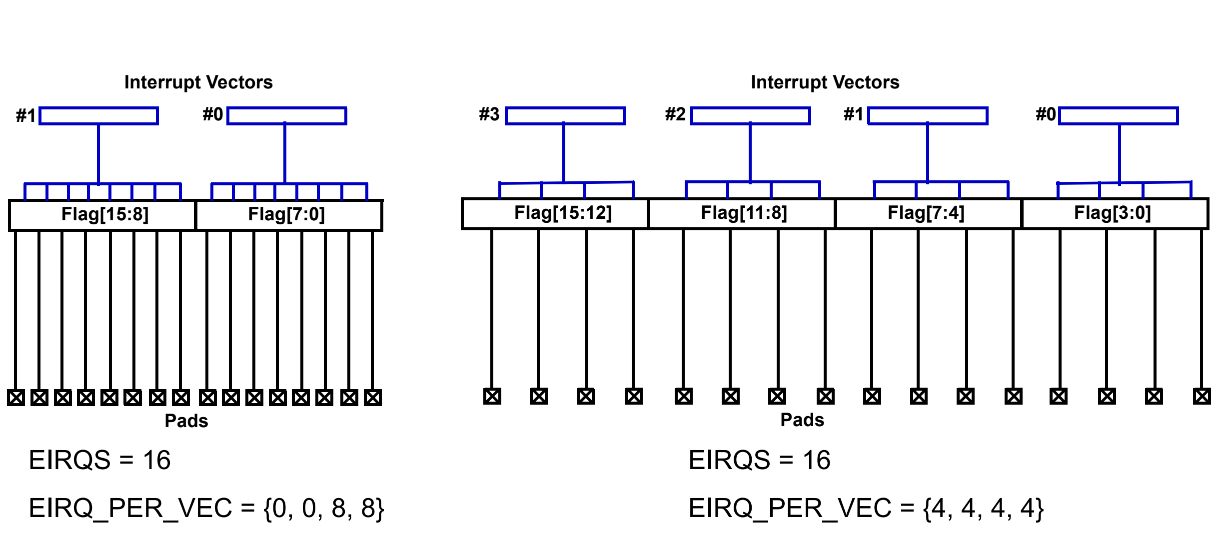
Note: Writing a '0' to both IREE[x] and IFEE[x] disables the external interrupt functionality for that pad completely (no system wakeup or interrupt will be generated on any activity on that pad)!

The active IRQ edge is controlled by the users through the configuration of the registers WIREER and WIFEER.

Each external interrupt supports an individual flag which is held in the flag register (WISR). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

For devices with less than the maximum number of interrupt sources, it is possible for the SoC to implement either all four interrupt vectors with fewer sources or to reduce the number of vectors as the number of sources are reduced. *Figure 1633* shows two examples of how this could be supported.

Figure 1633. Example of different interrupt to vector mapping at SOC level



80.4.4 On-Chip wakeups

The Wakeup unit supports up to 32 on-chip wakeup sources. It combines the on-chip wakeups with the external ones to generate a single wakeup to the system.

Note: If WUPS = 0, the unused wakeup_in input has no effect on the sys_wakeup output.

80.4.4.1 On-Chip wakeup management

In order to allow software to determine the wakeup source at one location, on-chip wakeups are reported along with external wakeups in the WISR register (see *Figure 1625* for details). Enabling and clearing of these wakeups are done via the on-chip wakeup source's own registers.

81 Real Time Clock / Autonomous Periodic Interrupt (RTC/API)

81.1 Overview

The RTC is a free running counter used for time keeping applications. The RTC may be configured to generate an interrupt at a predefined interval independent of the mode of operation (run mode or low power mode). When the RTC interval is reached, the RTC will generate a wakeup request to the wakeup unit. The RTC can be configured to generate an interrupt on the RTC interval timeout.

The RTC also supports an autonomous periodic interrupt (API) function which can be used to generate a periodic event to the wakeup unit or an interrupt request.

81.2 Features

Features of the RTC/API include:

- 3 selectable counter clock sources:
  - LPRC prescaled by 8 (128 kHz)
  - SXOSC (32 kHz)
  - IRCOSC (16 MHz)
- Clock sources can optionally be prescaled by 512 or 32
- 32-bit counter
  - Supports times up to 1.5 months with 1 ms resolution
  - Runs in all modes of operation, including normal RESET
  - Reset when disabled by software and by POR
- 32-bit compare value to support an extended range of interrupt intervals
- RTC compare value changeable while counter is running
- RTC status and control register are reset only by POR
- Autonomous periodic interrupt (API) to support an extended range of interrupt intervals
- Configurable interrupt for RTC match, API match, and RTC rollover
- Configurable wakeup event for RTC match and API match

Figure 1634. RTC/API block diagram

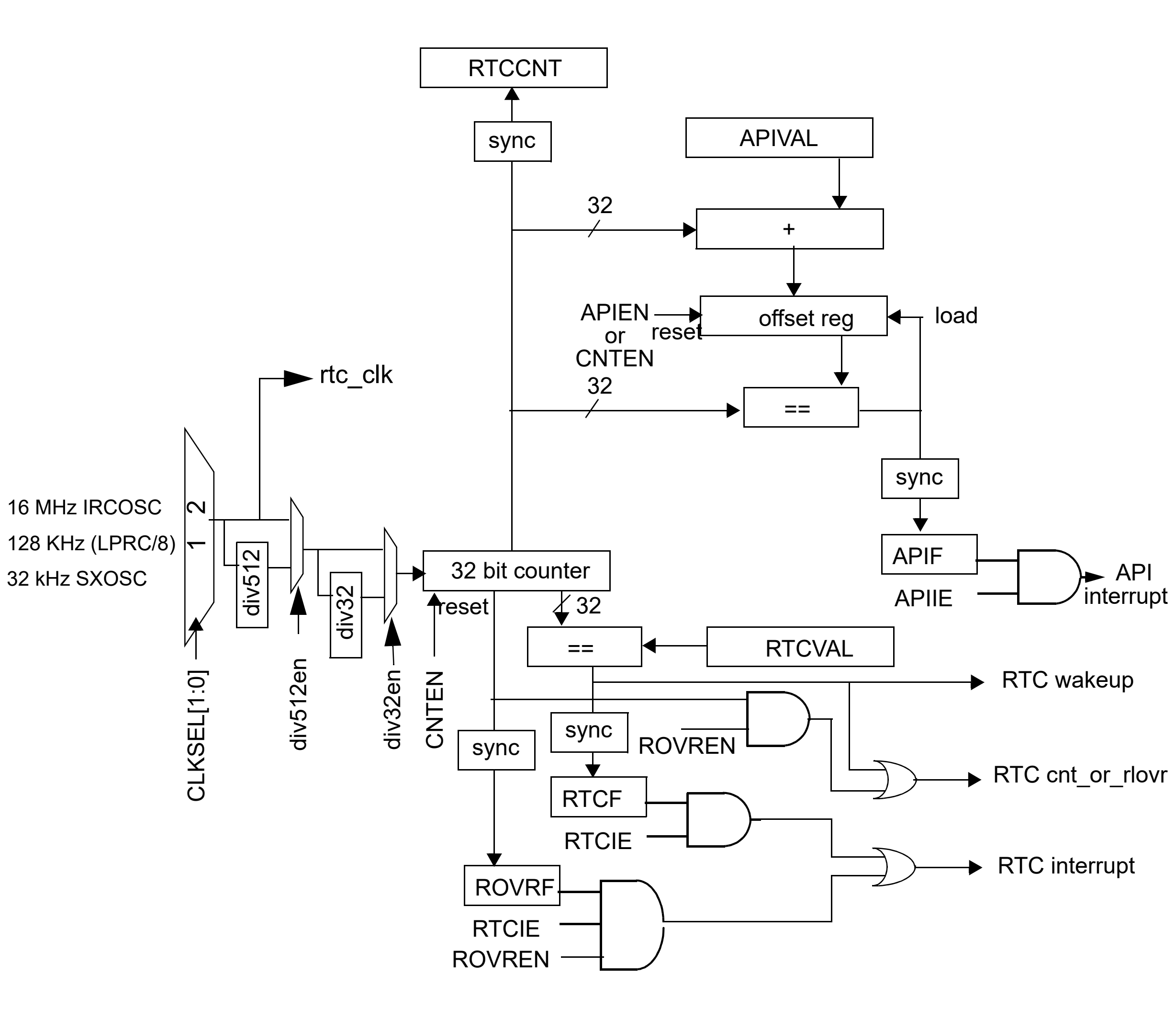
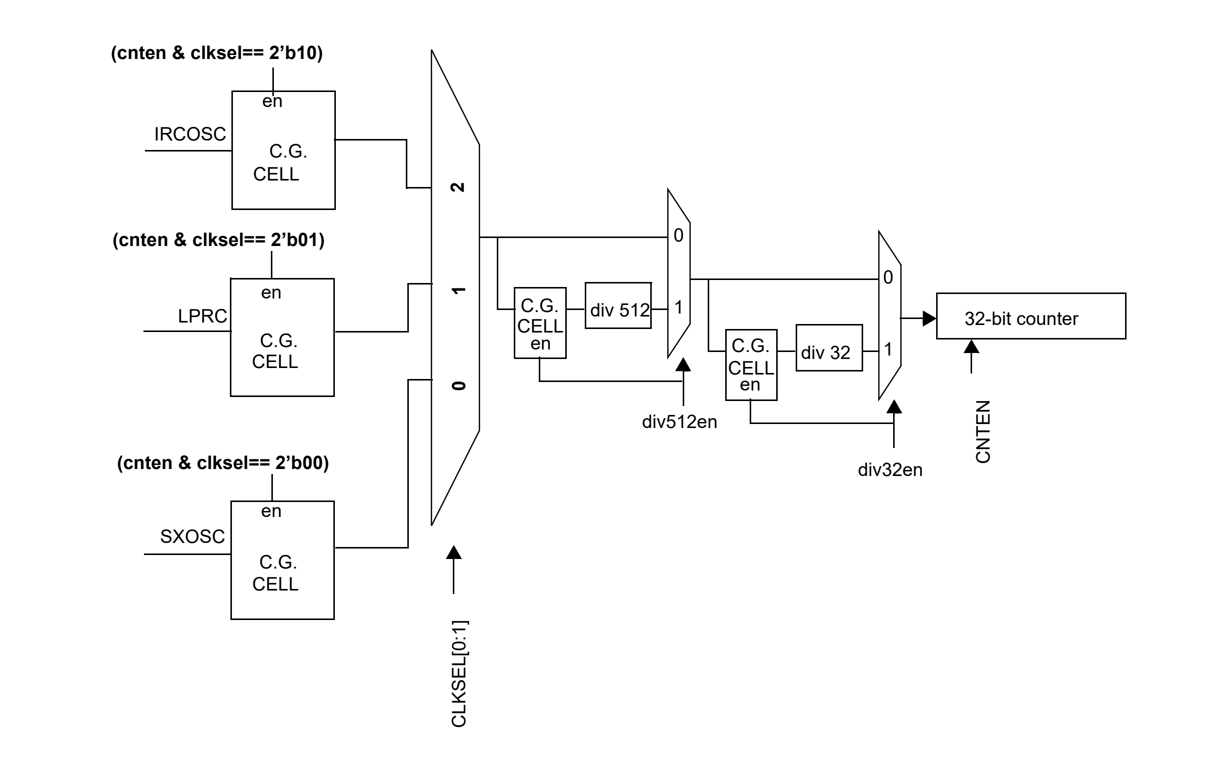


Figure 1635. Clock gating for RTC clocks



81.3 Modes of operation

81.3.1 Functional mode

There are two functional modes of operation for the RTC: normal operation and low power mode. In normal operation, all RTC registers can be read or written. The RTC/API and associated interrupts are optionally enabled. In low power mode, the bus interface is disabled and no configuration changes are permitted. The RTC/API is enabled if enabled prior to entry into low power mode.

81.3.2 Debug mode

If RTCC[FRZEN] is set, the counter will stop on the last valid count when the device enters debug mode. On debug mode exit, the counter will resume counting from the frozen value. If RTCC[FRZEN] is clear, the counter will continue counting (and set flags accordingly) when the device is in debug mode.

81.4 Register descriptions

The registers listed in *Table 1593* are described in the following sections.

Table 1593. RTC/API register map

Address offset	Register	Location
0x0000	RTC Supervisor Control Register (RTCSUPV)	*Section 81.4.1*
0x0004	RTC Control Register (RTCC)	*Section 81.4.2*
0x0008	RTC Status Register (RTCS)	*Section 81.4.3*
0x000C	RTC Counter Register (RTCCNT)	*Section 81.4.4*
0x0010	API Compare Value Register (RTC_APIVAL)	*Section 81.4.5*
0x0014	RTC Compare Value Register (RTC_RTCVAL)	*Section 81.4.6*

81.4.1 RTC Supervisor Control Register (RTCSUPV)

The RTCSUPV register contains the SUPV bit which determines whether other registers are accessible in supervisor mode or user mode.

Note: RTCSUPV register is accessible only in supervisor mode.

Offset: 0x0000 Access: Supervisor R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SUPV	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1636. RTC Supervisor Control Register (RTCSUPV)

Table 1594. RTCSUPV register field descriptions

Field	Description
0 SUPV	RTC Supervisor Bit 0 All registers are accessible in both user as well as supervisor mode. 1 All other registers are accessible in supervisor mode only.

81.4.2 RTC Control Register (RTCC)

The RTCC register contains:

- RTC counter enable
- RTC interrupt enable
- RTC clock source select
- RTC compare value
- API enable
- API interrupt enable
- API compare value

Offset: 0x0004 Access: User R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNTEN	RTCIE	FRZEN	ROVREN	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APIEN	APIIE	CLKSEL		DIV512EN	DIV32EN	0	0	0	0	0	0	0	0	0	TRIG_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1637. RTC Control Register (RTCC)

Table 1595. RTCC register field descriptions

Field	Description
0 CNTEN	Counter Enable The CNTEN bit enables the RTC counter. Clearing CNTEN has the effect of asynchronously resetting (synchronous reset negation) all the RTC and API logic as well as resetting the 32-bit counter. This allows for the RTC configuration and clock source selection to be updated without causing synchronization issues. 0 Counter disabled and reset 1 Counter enabled
1 RTCIE	RTC Interrupt Enable The RTCIE bit enables interrupts requests to the system if RTCF is asserted. 0 RTC interrupts disabled 1 RTC interrupts enabled
2 FRZEN	Freeze Enable Bit If RTCC[FRZEN] is set, the counter will stop on the last valid count when the device enters debug mode. On debug mode exit, the counter will resume counting from the frozen value. 0 Counter running in debug mode 1 Counter stops (freezes) in debug mode
3 ROVREN	Counter Roll Over Interrupt Enable The ROVREN enables interrupt requests when the RTC has rolled over from 0xFFFF_FFFF to 0x0000_0000. The RTCIE bit must also be set in order to generate an interrupt from a counter rollover. 0 RTC rollover interrupt disabled 1 RTC rollover interrupt enabled
16 APIEN	Autonomous Periodic Interrupt Enable The APIEN bit enables the autonomous periodic interrupt function. 0 API disabled 1 API enabled

Table 1595. RTCC register field descriptions (continued)

Field	Description
17 APIIE	API Interrupt Enable The APIIE bit enables interrupts requests to the system if APIF is asserted. 0 API interrupts disabled 1 API interrupts enabled
18:19 CLKSEL	Clock Select The CLKSEL[0:1] bits select the clock source for the RTC. CLKSEL may only be updated when CNTEN is 0. The user should ensure that oscillator is enabled before selecting it as a clock source for RTC. 00 SXOSC 01 LPRC prescaled by 8 10 IRCOSC
20 DIV512EN	Divide by 512 enable The DIV512EN bit enables the 512 clock divider. DIV512EN may only be updated when CNTEN is 0. 0 Divide by 512 is disabled. 1 Divide by 512 is enabled.
21 DIV32EN	Divide by 32 enable The DIV32EN bit enables the 32 clock divider. DIV32EN may only be updated when CNTEN is 0. 0 Divide by 32 is disabled. 1 Divide by 32 is enabled.
31 TRIG_EN	Trigger enable Used for self clearing of WKPU API event. This bit is required in case API event is being used to trigger SSWU eCTU commands. 0 Trigger enable is disabled. 1 Trigger enable is enabled.

81.4.3 RTC Status Register (RTCS)

The RTCS register contains:

- RTC interrupt flag
- API interrupt flag
- ROLLOVR Flag

Offset: 0x0008 Access: User R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RTCF	0	0	0	0	0	0	0	0	0	0	INV_RTC	INV_API	INV_ANL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	APIF	0	0	ROVRF	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1638. RTC Status Register (RTCS)

Table 1596. RTCS register field descriptions

Field	Description
2 RTCF	RTC Interrupt Flag The RTCF bit indicates that the RTC counter has reached the counter value matching RTCVAL. RTCF is cleared by writing a 1 to RTCF. Writing a 0 to RTCF has no effect. 0 RTC counter is not equal to RTCVAL. 1 RTC counter matches RTCVAL.
13 INV_RTC	Invalid RTC write This bit becomes value 1 after a value has been written to the RTCVAL register and the synchronization process is continuing. During this synchronization period, any attempt to write to the RTCVAL register again will be ignored.
14 INV_API	Invalid APIVAL write This bit becomes value 1 after a value has been written to the APIVAL register and the synchronization process is continuing. During this synchronization period, any attempt to write to the APIVAL register again will be ignored.
15 INV_ANL	Invalid ANLCMP write This bit becomes value 1 after a value has been written to the ANLCMP_CNT register and the synchronization process is continuing. During this synchronization period, any attempt to write to the ANLCMP_CNT register again will be ignored.
18 APIF	API Interrupt Flag The APIF bit indicates that the RTC counter has reached the counter value matching API offset value. APIF is cleared by writing a 1 to APIF. Writing a 0 to APIF has no effect. 0 No API interrupt 1 API interrupt
21 ROVRF	Counter Roll Over Interrupt Flag The ROVRF bit indicates that the RTC has rolled over from 0xffff_ffff to 0x0000_0000. ROVRF is cleared by writing a 1 to ROVRF. 0 RTC has not rolled over. 1 RTC has rolled over.

81.4.4 RTC Counter Register (RTCCNT)

The RTCCNT register contains the current value of the RTC counter.

Offset: 0x000C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RTCCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RTCCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1639. RTC Counter Register (RTCCNT)

Table 1597. RTCCNT register field descriptions

Field	Description
0:31 RTCCNT	RTC Counter Value Due to the clock synchronization, the RTCCNT value may actually represent a previous counter value.

81.4.5 API Compare Value Register (RTC_APIVAL)

The APIVAL bits are compared to the RTC counter bits and if a match occurs, an interrupt/wakeup request is asserted.

Offset: 0x0010 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	APIVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	APIVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1640. API Compare Value Register (RTC_APIVAL)

Table 1598. RTC_APIVAL register field descriptions

Field	Description
0:31 APIVAL	API Compare Value The APIVAL bits are compared to the RTC counter bits and if a match occurs, an interrupt/wakeup request is asserted. **Note:** API functionality is active only when APIVAL is non zero. The first API interrupt takes two more cycles because of synchronization of APIVAL to rtc clock. After that, interrupts are periodic in nature and it takes APIVAL+1 cycles. Minimum supported value of APIVAL is 4. This is due to synchronization issues.

81.4.6 RTC Compare Value Register (RTC_RTCVAL)

The RTCVAL bits are compared to the RTC counter bits and if a match occurs, RTCF is set.

Note: If RTC interrupt is enabled (RTCIE), it is required after each RTC counter match to reload the RTCVAL via software, to enable the interrupt again.

With RTCC[CTNEN] = 1 (RTC running) the user needs to wait the RTCS[RTC_INV] goes to zero (completed the synchronization process) before updating the RTCVAL counter. Otherwise the latter update will have no effect.

Offset: 0x0014 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RTCVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RTCVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1641. RTC Compare Value Register (RTC_RTCVAL)

Table 1599. RTC_RTCVAL register field descriptions

Field	Description
0:31 RTCVAL	RTC Compare Value The RTCVAL bits are compared to the RTC counter bits and if a match occurs, RTCF is set. **Note:** Minimum supported value of RTCVAL is 4. This is due to synchronization issues.

81.4.7 RTC Analogue Comparator Count Register (RTC_ANLCMP_CNT)

This register contains the number of API clocks corresponding to the trigger_en_out_pulse to be generated.

Offset: 0x0018 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ANLCMP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	ANLCMP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1642. RTC Analogue Comparator Count Register (RTC_ANLCMP_CNT)

Table 1600. RTC_ANLCMP_CNT register field descriptions

Field	Description
0:31 ANLCMP	The trigger_en_out pulse will be generated (after the generation of wakeup_api signal) as per the number of API clocks programmed into this register.

81.5 RTC functional description

The RTC consists in a 32-bit free running counter enabled with the RTCC[CNTEN] bit. (When CNTEN is cleared, the counter is asynchronously reset.) The value of the counter may be read via the RTCCNT register. Note that due to the clock synchronization, the RTCCNT value may actually represent a previous counter value. The difference between the counter and the read value depends on ratio of counter clock and main clock. Maximum possible difference between the two is 6 count values.

The clock source to the counter is selected with the RTCC[CLKSEL] field, which gives four options for clocking the RTC/API. The three clock sources are:

- LPRC prescaled by 8 (128 kHz)
- SXOSC (32 kHz)
- IRCOSC (16 MHz)

The output of the clock mux can be optionally divided by combination of 512 and 32 to give a 1 ms RTC/API count period for different clock sources. Note that the RTCC[CNTEN] bit must be disabled when the RTC/API clock source is switched.

When the counter value matches the 32-bit value in the RTCC[RTCVAL] field, then the RTCS[RTCF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[RTCIE] interrupt enable bit is set, then the RTC interrupt request is generated. If there is a match while in low power mode then the RTC will first generate a wakeup request to force a wakeup to run mode, then the RTCF flag will be set.

A rollover interrupt can be generated when the RTC transitions from a count of 0xFFFF_FFFF to 0x0000_0000. Rollover events are enabled by setting the RTCC[ROVEN] bit. If this bit is set, an RTC rollover will cause a request. To enable interrupts on a rollover request, the RTCC[RTCIE] bit also needs to be set.

All the flags and counter values are synchronized with ipg_clk the main clock. It is assumed that ipg_clk the main clock frequency is always more than or equal to the rtc_clk RTC clock used to run the counter.

81.6 API functional description

Setting the RTCC[APIEN] bit enables the autonomous interrupt function. The 32-bit RTCC[APIVAL] field selects the time interval for triggering an interrupt and/or wakeup event. Since the RTC is a free running counter, the APIVAL is added to the current count to calculate an offset. When the counter reaches the offset count, a interrupt and/or wakeup request is generated. Then the offset value is recalculated and re-triggers a new request when the new value is reached. APIVAL may only be updated when APIEN is disabled. When a compare is reached, the RTCS[APIF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[APIIE] interrupt enable bit is set, then the API interrupt request is generated. If there is a match while in low power mode, then the API will first generate a wakeup request to force a wakeup into normal operation, then the APIF flag will be set.

Appendix A Acronyms and abbreviations

Table 1601 lists acronyms and abbreviations used in this document.

Table 1601. Acronyms and abbreviations

Term	Meaning
AHB	Advanced high-performance bus
AIC	PBRIDGE (AIPS) integrity checker
AMBA	Advanced microcontroller bus architecture
APFC	Auxiliary pin function control
APU	– Advanced protocol unit – Auxiliary processing unit
ASILB	Automotive SIL B
AUTOSAR	Automotive Open System Architecture
AVB	Audio Video Bridging
BAF	Boot assist flash
BCU	Buffer control unit
BIST	Built-in self test
BIU	Bus interface unit
BMIF	Bus master interface
CAN	Controller area network
CC	Communication controller
CDC	Clock domain crosser
CF	Critical fault
CGM	– Clock generation module – Conditional group member
CGMC	Conditional group membership count
CHI	Controller host interface
CID	Controller ID
CJTAG	Compact JTAG
CLTAPC	Chip-level TAPC
CP	Check packet
CPA	Check process active
CMU	Clock monitoring unit
CPU	Central processing unit
CQM	Clock quality monitor
CRC	Cyclic redundancy check

Table 1601. Acronyms and abbreviations (continued)

Term	Meaning
CSI	Combined serial interface. DSPI configuration that alternates DSI and SPI frames.
CTU	Cross triggering unit
DCF	Device configuration format
DCU	Data channel unit
DMA	Direct memory access
DMAMUX	Direct memory access controller multiplexer
DSPI	Deserial serial peripheral interface
DSI	Deserial serial interface. DSPI configuration that serializes and deserializes registers or signals to allow for pin reduction.
EBI	External bus interface
ECC	Error correction code
ECSM	Error correction status module
EIM	Error injection module
EMI	Electromagnetic interference
EOT	End of transfer
EOUT	Error out
EPU	Extended protocol unit
ESD	Electromagnetic static discharge
FCCU	Fault collection and control unit
FMPLL	Frequency-modulated phase-locked loop
FOSU	FCCU output supervision unit
FPEC	Flash program erase controller
FSM	Finite state machine
FSS	Frame start sequence
FTTI	Fault tolerant time interval
GPIO	General-purpose I/O
HIF	Host interface
HVD	High voltage detector
HW	Hardware
ID	LIN identifier field
IEEE	Institute of Electrical and Electronics Engineers
INTC	Interrupt controller
INTF	Interface
IP	Intellectual propriety

Table 1601. Acronyms and abbreviations (continued)

Term	Meaning
IPS	– Internal peripheral system – IPI slave bus
IRQ	Interrupt request
ISR	Interrupt service routine
JDC	JTAG data communication
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
Jscan	JTAG scan
LIN	Local interconnect network
LINFlexD	Local interconnect network controller
LRAM	Look up table RAM
LUT	Look up table
LVD	Low voltage detector
LWB	Late-write buffer
MB	Message buffer
MBIDX	Message buffer index
MBIST	Memory built-in self test
MBM	Message buffer management
MBNum	Message buffer number
MC	MC_CGM, MC_ME, MC_PCU, and MC_RGM modules
MCU	Microcontroller unit
ME	Mode entry module (MC_ME)
MEMU	Memory error management unit
MFD	Multiplication factor divider
Mscan	Maximum-flexibility scan
MT	Macrotick
MTS	Media access test symbol
Mux	Multiplex
NCF	Non-critical fault
NIT	Network idle time
NMI	Non-maskable interrupts
nTRST	Active low test reset
NVM	Non-volatile memory
OPP	Over-program protection
OSC	Oscillator

Table 1601. Acronyms and abbreviations (continued)

Term	Meaning
Oscan	Optimized scan
OTP	One time programmable
PASS	Password and device security module
PBRIDGE	Peripheral bridge
PCM	Platform configuration module
PCU	Power control unit
PE	Protocol engine
PFLASH	Platform flash controller
PIT	Periodic interrupt timer
PLL	Phase-locked loop
PMC	Power management controller
POC	Protocol operation control
PRAM	Platform RAM controller
PSS	Pause selection state
PWD	Password
RC	– Resistance-capacitance – Resistor-capacitor
RCC	Redundancy control checkers
RCCU	Redundancy control checker unit
RCOSC	RC oscillator
RF	Recoverable fault
RFD	Reduced frequency divider
RGM	Reset generation module
RMW	Read-modify-write
RSU	Reset and selection unit
RTL	Register transfer language
RWW	Read-while-write
RX	Receive
SAG	Safety application guide
SAR ADC	Successive approximation register analog-to-digital converter
SECDED	Single-bit error correction, double-bit error detection
SEQ	Sequencer engine
SIL	Safety integrity level
SIUL	System integration unit lite
SMPU	System memory protection unit

Table 1601. Acronyms and abbreviations (continued)

Term	Meaning
SoC	System-on-chip
SOR	Sphere of redundancy
SP	Scan packet
SPA	Scan packet active
SPI	Serial peripheral interface
Sscan	Segmented scan
SSCM	System status and configuration module
SSD	Scan selection directive
SSM	Scan state machine
STCU	Self-test control unit
STL	System test logic
STM	System timer module
SW	Software
SWT	Software watchdog timer
TAP	Test access point
TAP.1	An IEEE 1149.1 test access port
TAP.7	An IEEE 1149.7 test access port
TBD	To be defined
TCA	Tap controller access
TCD	Transfer control descriptor
TCK	Test clock
TCKC	Test clock compact
TCU	Time control unit
TDI	Test data input
TDIC	Test data input compact
TDO	Test data output
TDOC	Test data output compact
TMS	Test mode select
TMSC	Test mode select compact
TS	Target System
TSENS	Temperature sensor
TX	Transmit
UART	Universal asynchronous/synchronous receiver transmitter
UF	Unrecoverable fault
VCO	Voltage controlled oscillator

Table 1601. Acronyms and abbreviations (continued)

Term	Meaning
VREG	Voltage regulator
WDG	Watchdog timer
WKPU	Wakeup unit
WS	Wait state
XBAR	Crossbar
XBIC	XBAR integrity checker
ZBS	Zero bit scan
µT	Microtick

Revision history

Table 1602. Document revision history

Date	Revision	Changes
26-Jul-2017	1	Initial release
25-Apr-2018	2	Following are the changes in this version of the Reference manual: *Section : Overview* on cover page – Removed text “or symmetrical dual”. – Updated operating speed from 180 MHz to 120 MHz. *Chapter 2: Introduction* *Section 2.1: SPC584Bx microcontroller*: Updated the section. *Chapter 5: Memory Map* *Table 5: Flash memory map* Data Flash Blocks: removed “FMC” from the description for all addresses *Table 6: System and stand-by RAM memory map*: Replaced HSM memory spaces by a cross-reference to HSM memory map table. Updated HSM end offset. Updated *Table 8: HSM memory map* *Chapter 6: Functional safety* *Table 12: PMHF module categories*: Updated definition of Non-safety modules *Section 6.4.3.2: All-x words and ECC*: – Added sentence starting with “Memories that include addresses in the ECC...” – Reworked the paragraph starting with “In case of not valid codeword...” *Section 6.4.3.3: ECC failure handling*, *Section 6.4.3.4: Memory Error Management Unit (MEMU)* and *Section 6.4.3.4.1: Interface to ECC units*: Added “with same (optional) syndrome” *Table 13: Module classification*: Moved Concentrator DMA from NoSaMos to ViMos *Table 14: FCCU failure inputs* – Added fault 48. – Added event in failure descriptions 49 and 50. – Removed faults 115-123 not available for users. *Table 15: ECC RAM implementations*: Removed I-MEM Memory for Core_2 *Chapter 7: Device configuration* Made extensive changes in this chapter. *Chapter 8: Reset and Boot* *Section 8.3.3*, *Section 8.3.4*, *Section 8.3.5*, *Section 8.3.8* and *Section 8.3.9*: Deleted all indications of duration of each phase (“This phase is completed in...”) *Chapter 9: Device Configuration Format (DCF) Records* *Table 107: DCF client list*: Added “User serial boot”.

Table 1602. Document revision history (continued)

Date	Revision	Changes
25-Apr-2018	2 (cont'd)	*Chapter 10: Power management* *Figure 45: Device power management framework*. Replaced Core 0 with Core 2 and SRAM size for PD3 to 96 KB. *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* *Section 11.3: Architecture*: Added the paragraph starting with "In STANDBY mode, only RTC is powered on,..." *Chapter 16: System Integration Unit Lite2 (SIUL2)* *Table 205: SIUL2_MSCR_IOn field description*: – Changed description of ILS bitfield for 00 value from "Automotive" to "Reserved". – ILS bits value 00 is Reserved. *Figure 116: SIUL2 MCU ID Register #1 (SIUL2_MIDR1)*, in the footnote removed the default value of PKG bitfiled. *Section 16.2.2.12: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IOn)*, replaced "input/pull-up enabled" with "ZiH". *Figure 127: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IOn)*; ILS bit field Reset value is 0b11. *Chapter 20: System Memory Protection Unit (SMPU)* *Section 20.7: Application information*; Dual core application example removed. *Figure 156: Control/Error Status Register 1 (CESR1)*; Reset value for CH02M is 0b0100. *Chapter 24: DMA channel multiplexer (DMACHMUX)* Updated *Figure 217: DMACHMUX triggered channels* *Chapter 25: Clocking* *Table 302: MC_CGM relationship to clocks*, – Deleted "Bridge clock" from Output clock column of CGM_SC_DC1. – Reviewed description of registers CGM_AC12_DC1/2/3. *Table 304: Maximum auxiliary level clock frequencies*, update "Nominal programmed divider max output frequency" value of "Aux Clock 6 Selector". *Figure 220: Clock generation* Added information in Note 3. *Chapter 26: Dual PLL digital interface (PLLDIG)*: *Table 316: PLL0CR field descriptions*: EXPEDIE bitfield description enhanced *Table 318: PLL0DV field descriptions* and *Table 321: PLL1DV field descriptions*: Updated division factors of RFDPHI. *Section 26.6.3: Frequency modulation*, replaced PLLnCR[MFD] with PLL1DV[MFD]. *Section 26.7: Initialization information*: added footnote linked to "procedure". Changed "MC_ME_<mode>" by "<ME_mode>, "MC_ME_" or "ME" by "<ME>" and "MC_CGM_ACn" by "<CGM_ACn>" throughout all this chapter.

Table 1602. Document revision history (continued)

Date	Revision	Changes
25-Apr-2018	2 (cont'd)	*Chapter 27: Clock Monitor Unit (CMU)* *Table 326: CMU_CSR field descriptions* – SFM: added sentences "MDR[MD] ... CSR[SFM]=1" and "Software ... bit". – RCDIV: added sentence "Ensure ... [RCDIV]". – CME: added sentence "CMU_HFREF ... (CSR[CME=1])". *Section 27.5.2: CLKMN0_RMT supervisor*: deleted Note "f_{CLKMN0_RMT} ... *monitoring*". *Chapter 31: LPRC Digital Interface* *Section 31.2: Low Power RC oscillator*, deleted last paragraph concerning LPRCTRIM. *Section 31.3.1.1: Low Power RC Control register (CTL)*, deleted LPRCTRIM bitfield. *Chapter 38: Analog-to-Digital Converters (ADC) Configuration* *Section 38.2.1.1.2: 12-bit SAR ADC B0*: Removed references to SUPERVISOR. *Section 38.2.1: Successive Approximation Register Analog-to-Digital Converter (SAR ADC)*; Removed SUPERVISOR refered to 12 bit SAR ADC. *Chapter 39: Successive Approximation Register Analog-to-Digital Converter (SARADC)* *Figure 382: SARADC block diagram*, removed "watchdog trigger". *Section 39.3.1: Main features*: – Deleted "Trigger outputs on watchdog threshold crossover events" only for *WDG_trigout_yes.* – Deleted the bullet "Watchdog thresholds crossover". *Section 39.4.5.1: CTU trigger mode*: Replaced JABORT by JABORTCHAIN (was a typo). *Section 39.4.8: Programmable analog watchdog*: Deleted the latest paragraph about "external trigger output port which". *Section 39.5.1.1: Main Configuration Register (MCR)*: Deleted WTRIGOUT bitfiled. *Table 465: CTU modes*: Added comment in the footnote about CTSEN *Table 468: MCR field descriptions*: Replaced SDADC by SARADC in FRZ bitfield description (was a typo). *Chapter 43: Software Watchdog Timer (SWT)* *Section 43.1.1: Overview*: Added Note "The SWT runs over all device ...". *Chapter 44: Periodic Interrupt Timer (PIT)* *Section 44.2.1.6: PIT RTI Timer Control Register (RTI_TCTRL)*: Removed bitfield CHN.

Table 1602. Document revision history (continued)

Date	Revision	Changes
25-Apr-2018	2 (cont'd)	*Chapter 50: LINFlexD* *Figure 960: Master node: DMA Tx FSM (concept scheme)* and *Figure 964: Slave node: DMA Tx FSM (concept scheme)*: Added note "DTF Flag gets auto cleared by Hardware if Software does not clear it within 3 ipg_clk cycles.". *Figure 979: LIN Time-Out Control Status Register (LINTCSR)*: Updated HTO reset value by adding a note (1) on bits 26:27. *Figure 985: LIN Control Register 2 (LINCR2)*: Updated IOPE reset value note (3). *Chapter 51: Reset Generation Module (MC_RGM)* *Figure 1053: Peripheral Reset Status Register 7 (RGM_PSTAT7)*, *Table 1071: RGM_PRST7 register field descriptions*: Fixed field name ETHERNET_0_STAT instead of 'PSTAT'. *Chapter 52: Boot Assist Flash (BAF)* *Section 52.3.7: Optionally perform a serial boot*: Updated introduction paragraph. *Chapter 53: System Status and Configuration Module (SSCM)* Updated *Table 1104: Life Cycle slots*, *Table 1105: Life Cycle Slots in Memory* and *Table 1106: Resulting Life Cycle*. Removed "Glossary" section. *Chapter 54: Power management controller digital interface (PMC_Dig)* Updated the access type of all register descriptions as per *Section 54.3: IPS bus interface* "The registers of the PMC_Dig block are accessible (read/write) in each access mode: user, supervisor or test". *Table 1107: Power Management Controller Memory Map*: Added PNREG_S register. *Chapter 56: Mode Entry Module (MC_ME)* *Table 1178: ME_<mode>_MC Field Descriptions*: FLAON bitfield is valid for both Data and Instruction cache *Chapter 62: Sequence Processing Unit (SPU)* For each occurrence of "Calibration and Debug" in the chapter, added "(or Debug and Trace)" *Chapter 66: LVDS Fast Asynchronous Serial Transmission – High Speed Debug* *Table 1326: PLLCR field descriptions*: Updated description of FBDIV

Table 1602. Document revision history (continued)

Date	Revision	Changes
25-Apr-2018	2 (cont'd)	*Chapter 73: Memory Error Management Unit (MEMU)* *Section 73.1: Introduction*: Replaced "Peripheral System RAM" by "Peripheral RAM" in the second bullet. *Table 1452: SYS_RAM_UNCERR_ADDR field descriptions*, *Table 1457: PERIPH_RAM_UNCERR_ADDR field descriptions* and *Table 1462: FLASH_UNCERR_ADDR field descriptions*: Added the sentence "This register must be READ ONLY when its corresponding valid bit is not asserted." to ERR_ADD field description. *Chapter 75: Fault Collection and Control Unit (FCCU)* *Table 1488: FCCU_CFG field descriptions*: updated description of bitfield FCCU_SET_CLEAR. *Figure 1533: FCCU CTRL Key Register (FCCU_CTRLK)* and *Table 1487: FCCU_CTRLK field descriptions*: Changed CTRLK field name to CTRLK_VAL. *Figure 1538: FCCU RF Key register (FCCU_RFK)* and *Table 1495: FCCU_RFK field descriptions*: Changed RFK field name to RFK_VAL. *Figure 1552: FCCU XTMR Register (FCCU_XTMR)* and *Table 1512: FCCU_XTMR field descriptions*: Changed XTMR field name to XTMR_VAL. *Figure 1556: FCCU Delta T Register (FCCU_DELTA_T)* and *Table 1517: FCCU_DELTA_T field descriptions*: Changed XTMR field name to XTMR_VAL. *Chapter 76: Self-Test Control Unit (STCU2)* *Table 1524: STCU2 memory map*: Renamed table title (was register set). *Section 76.5.8: STCU2 CRC Expected Status Register (STCU_CRCE)*: Removed equation number. *Figure 1572: STCU2 Error FM Register (STCU_ERR_FM)*: Set reset value of bits 0:15 to all 0. Removed section "Register map" *Chapter 78: Password and Device Security Module (PASS)* *Table 1552: LCSTAT register field descriptions*: Updated description of LIFE bitfield with value 110 (ST production) *Figure 1599: Password Group n - Lock 0 Status register (LOCK0_PGn)*: Added ATSL field at bit 1 and set undefined reset with figure note. *Table 1556: LOCK0_PGn register field descriptions*: Added ATSL field and its description at bit 1. *Section 78.2.1.4: Challenge Input register (CINn)*: Added the following sentence "Valid password criteria is that it should not be all Zeros or all Ones". *Chapter 93: Tamper Detection Module (TDM)* *Table 2346: TDM_STO_KEYn field descriptions*: Added "See the section...". *Figure 2446: Diary implementation*: Changed Diary_TDR0 size from 256x64 to 256x128 bits.

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3	Following are the changes in this version of the Reference manual: *Chapter 5: Memory Map* *Table 8: HSM memory map*: Updated the table. *Table 9: Peripheral memory map (PBRIDGE 2)* Updated size + end offset of MC_CGM and start offset of the following Reserved field. *Table 11: UTest memory map*: Added a footnote, “Factory could program........find it” to the “UTest DCF records” for Start offset of 0x0370. *Chapter 6: Functional safety* *Table 13: Module classification* - NoSaMos: Moved “Concentrator HSM” to ViMos classification. *Table 15: ECC RAM implementations*: Replaced all JTAG occurrences by ITAG *Section 6.4.3.4: Memory Error Management Unit (MEMU)*: Rephrased the first paragraph. *Chapter 7: Device configuration* Made extensive changes in this chapter. *Chapter 8: Reset and Boot* *Section 8.3.1: Power-on and the Reset Generation Module*: Removed “long external reset” from the last paragraph. *Section 8.3.8: PHASE1[FUNC] Phase: temporization and monitoring setup*: added the latest sentence “During this phase, test modules are available.” *Figure 35: Reset Generation Module Reset Sequence*: – Removed a block of “no self test” – Added footnote, “To specify that no self-tests are executed in this case” – Updated the figure Updated footnote, “No self-tests are executed.” *Chapter 9: Device Configuration Format (DCF) Records*: Removed “FlexRay” from the entire chapter. *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* *Section 11.6.4: OPC*: Added second and third bullets to the set of pads behavior *Chapter 15: Core (z420n3) Description* *Section 15.9: Local memories*: Added this section. *Section 15.10: End-to-End ECC support*: Added this section. *Chapter 16: System Integration Unit Lite2 (SIUL2)* *Figure 126: Soc Configuration Register 0 (SIUL2_SCR0)*: – Updated the bit [8] from “ESR0_ASSERT” to “Reserved”. – Added a note for bit [8], “This bit shall be left at '0' default value by application.”

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	*Section 16.2.2.12: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IOn)*: Changed the latest sentence by referring to the device I/O_Definition document. *Table 205: SIUL2_MSCR_IOn field description*: Changed WPDE and WPUE field descriptions. *Chapter 17: Crossbar switch (XBAR)* *Table 214: XBAR memory map*: added MASTER channel n (n = 0 to 7) title and XBAR_MGPCRn registers at 0x0800 +n*0x100 address offsets. *Table 215: XBAR_PRSn field descriptions*: Added description of bitfield M3. *Table 216: XBAR_CRSn field descriptions*: Updated the note of PARK field from “Only select master.......behavior may occur.” to “Only select master.......parking configuration.” *Section 17.3.3: Master General Purpose Control Register (XBAR_MGPCRn)*: added this section. *Chapter 19: Peripheral Bridge (PBRIDGE)* *Section 19.2.3: Peripheral Access Control Register (PACRx)*: Added paragraph “The PBRIDGE registers themselves are mapped into the PACR0 address space and can only be accessed in supervisor mode by trusted bus masters.” *Table 225: Peripheral bridge memory map*: Updated some reserved spaces versus table notes. *Chapter 23: Enhanced Direct Memory Access (eDMA)* *Figure 178: Control Register (eDMA_CR)*: Changed GRP1PRI field reset value to 2’b01. *Chapter 25: Clocking* *Section 25.1: Introduction*: Updated this section and deleted text “as a backup clock”. Updated *Section 25.4: Default clock configuration* “Synchronous IP blocks...PLL or oscillator failure.” Updated *Figure 223: Clock distribution* *Section 25.5.1.6: Loss of clock detection*: deleted the last sentence “Software may enable or disable an optional backup clock in the PLL”. *Section 25.8.1: Loss of PLL/XOSC clock*: Updated this section. *Section 25.8.2: Loss of IRCOSC clock*: Updated this section and deleted text “and backup”. *Table 302: MC_CGM relationship to clocks* – Removed row-headed CGM_AC2_DC0 – Updated row-headed CGM_AC12_DC2 CGM_AC12_SC *Table 304: Maximum auxiliary level clock frequencies*: Added note to EMIOS_CLK. *Chapter 26: Dual PLL digital interface (PLLDIG)* *Section 26.1: Introduction*: Added note with a cross-reference to Clocking chapter for chip-specific implementation details.

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	*Section 26.4: Modes of operation*: – Changed "the operating modes" by "the operating mode" – Changed "The various modes of operation for the PLL are" by "The mode of operation for the PLL is defined below" *Figure 229: Dual PLL digital interface block diagram*: – Updated this figure by adding a divider by 2 in PLL0 block. – "Backup Clock Selection" block is made blue and note added "Backup Clock Selection" block in blue is not implemented". *Chapter 27: Clock Monitor Unit (CMU)* *Section 27.4.1.1: CMU Control Status Register (CMU_CSR)*: Updated the description of RCDIV from, "Ensure clock.......CMU_CSR[RCDIV]." to "Ensure clock......otherwise the operation can lead to a spurious OLR event." *Table 326: CMU_CSR field descriptions*: Updated the descriptions of SFM and CME fields. *Section 27.4.1.5: CMU Interrupt Status Register (CMU_ISR)*: updated Note. *Chapter 28: Clock Generation Module (MC_CGM)* Updated *Section 28.3.1.37: Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)* Updated *Section 28.3.1.43: Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)* Removed Section "Divider jitter injection" *Section 28.4.3.1: Fractional divider details*: Added this section. *Chapter 29: OSC digital interface (XOSC)* *Figure 298: XOSC Control register (XOSC_CTL)*: Changed I_OSC field access type to R/w1c (was RW). Added *Section 29.2.4.1: Automatic Level Control (ALC)* *Chapter 31: LPRC Digital Interface* *Section 31.2: Low Power RC oscillator*: Updated the second paragraph starting with "Low power RC oscillator is enabled..." *Figure 301: Low Power RC Control register (CTL)*: Updated the reset value of S_LPRC field to '1'. *Chapter 34: Flash memory controller* *Section 34.5.13.1.1: e2eECC and Flash accesses*: added the latest sentence. *Section 34.5.14: e2eECC on data Flash accesses*: added the latest sentence. *Chapter 35: Embedded Flash Memory* *Section 35.3.2.1: Module Configuration Register (MCR)*: Updated EEE field description. *Table 407: MCR field descriptions*: – Added sentence "Note that EDC post ECC reporting is suppressed for Data Flash and HSM Data Flash" at EEE field description.

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	– Improved wording of DWEE in both for code and data memory. *Table 410: MCRE field descriptions*: – Updated description of fields N64KL and N16KL. – Removed the row of field "N256KH". Improved wording for bitfield ADDR in *Table 422: ADR field descriptions*. *Section 35.4.4.1: Error correction code*: Added sentence, "EDC post ECC reporting is suppressed for Data Flash and HSM Data Flash" at the end of the fourth sentence. Added *Section 35.4.4.1.1: Multi-bits error management (more errors than uncorrectable detection capability)*. Updated size of customer OTP area in*Table 405: UTEST block memory map* Sentence in *Section 35.4.1: Reset* has been highlighted with Warning style *Chapter 36: Flash Memory Programming and Configuration* *Section 36.6.2: Creating the tamper detect diary*: – The maximum size of Diary updated from 2 KB to 4 KB. – Multiplied by 2 KB is updated to 4 KB. – Updated the *Example 7: Creating the tamper detect diary* *Example 4: Overriding secure write protection*: Updated paragraph list 2. *Chapter 38: Analog-to-Digital Converters (ADC) Configuration* *Section 38.2.1.1.1: 12-bit Fast SAR ADC*: Updated "analog input channel" for each SARADC instances. *Chapter 39: Successive Approximation Register Analog-to-Digital Converter (SARADC)* *Table 467: SARADC digital interface register map*: Removed "Register classification" column. *Section 39.4.4.1: Conversion timings*: Changed SAMPSEL by CTSEL field name. *Table 468: MCR field descriptions*: added note to NSTART, JSTART and PWDN field descriptions. *Table 487: PDEDR field descriptions*: added latest paragraph in PDED field description. *Section 39.4.6: Test channel connection with internal analog channel*: removed temperature sensor and bandgap from the first sentence of the first paragraph. *Section 39.5.1.11: Watchdog Threshold Register n (WTHRHLRn)*: Replaced in 1st note MSB=0 bits numbers by LSB=0 bit numbers. *Chapter 44: Periodic Interrupt Timer (PIT)* *Section 44.2.1.6: PIT RTI Timer Control Register (RTI_TCTRL)*: Removed the bitfield CH. *Section 44.5: Example Configuration for the Lifetime Timer*: Updated *Equation 29:* From, "current_uptime = PIT_LTMR64H << 64" to "current_uptime = PIT_LTMR64H << 32" *Chapter 45: Enhanced Modular IO Subsystem (eMIOS)* *Table 569: EMIOSMCR field descriptions*: Updated note in GTBE field description.

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	*Chapter 46: CAN subsystem* *Table 595: TSCC field descriptions*: – TCP: Removed the note, "With CAN FD..... generation (TSS = '10')." – TSS: Added a new note, "In case of CAN FD..... frames are received." – TSS: Updated the value of 10 from "External timestamp counter value used" to "Reserved". *Table 596: TSCV field descriptions*: – TSC: Updated the content from , "The internal/ external......(both Rx and Tx)." to "The internal.....(both Rx and Tx)." – TSC: Removed, "When TSCC. TSS = '10', TSC reflects .... no impact." *Section 46.3.5.2.3: Data Bit Timing and Prescaler Register (DBTP)*: corrected time quanta max number in the first sentence. *Table 608: XIDFC field descriptions*: corrected max enumerated value of LSE field to 0x7F. *Table 613: RXF0C field descriptions*: corrected max enumerated value of F0WM and FOS fields to 0x7F. *Chapter 49: Deserial Serial Peripheral Interface (DSPI)* *Section 49.3.6: DSPI Status Register (DSPI_SR)*: changed reset value from 0xDEAD_BEEF to 0x0201_0000 *Table 957: DSPI_MCR field descriptions*: added note to HALT field description. *Chapter 50: LINFlexD* *Section 50.2.2: UART mode features*: Updated the 3 latest bullets. *Table 1022: TCD setting: UART - Tx mode*: Removed note at DADDR[31:0]. *Table 1023: TCD setting: UART - Rx mode*: Removed note at SADDR[31:0]. *Table 1024: LINFlexD memory map*: Removed "Register classification" column. *Chapter 51: Reset Generation Module (MC_RGM)* *Table 1067: RGM_PRST3 register field descriptions*: In SAR_ADC_12BIT_B_RST field, added note "In case BCTU_0 is used to trigger SAR_ADC_12BIT_B...." *Section 51.4.7: 'Destructive' reset escalation*: Added footnote to "each 'destructive' reset". *Chapter 52: Boot Assist Flash (BAF)* *Section 52.1: Introduction*: Updated the fourth paragraph. *Section 52.3: Functional description*: Removed section "Production disable activation protocol and implementation". *Section 52.3.7: Optionally perform a serial boot*: Deleted text "All Tx-channels have to be disabled during this phase"

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	*Chapter 53: System Status and Configuration Module (SSCM)* *Figure 1070: HSM and User Option Status Register (UOPS)* and *Table 1099: UOPS field descriptions*: – Added fields FL_RUN, BAF_FA and FL_STDBY and their respective descriptions. – Changed HSB bit field length to 3 bits and updated its description. *Section 53.3.3: Boot mode functionality*: Updated Standby-Boot section. *Chapter 54: Power management controller digital interface (PMC_Dig)* Removed “FlexRay” from the entire chapter. *Section 54.5.27: Miscellaneous CTRL REG (MISC_CTRL_REG)*: – Changed Bit 31 to RESERVED with a footnote. – Changed Bit 29 to Reserved – Added a footnote to Bit 28 *Table 1111: REE_LV0 field descriptions* and *Table 1121: REE_HV0 field descriptions*: Changed “REE” by “RES” in REE_XX field descriptions. *Table 1112: RES_LV0 field descriptions* and *Table 1121: REE_HV0 field descriptions*: Changed “RES” by “REE” in RES_XX field descriptions. *Section 54.5.53: User BIST Time1 and Time0 Register (BIST_TIME10)*: Changed Bit 3 and Bit 19 to RESERVED. *Section 54.5.54: User BIST Time3 and Time2 Register (BIST_TIME32)*: Changed Bit 3 and Bit 19 to RESERVED. *Section 54.5.55: User BIST Time6 and Time5 Register (BIST_TIME65)*: Changed Bit 3 and Bit 19 to RESERVED. *Section 54.9: Flash interface (DCF client usage)*: Added 2 paragraphs relative to monitors reactions. *Section 54.10: Voltage monitor BIST programming*: Updated this section. *Chapter 56: Mode Entry Module (MC_ME)* *Section 56.4.1: Mode transition request*: – Added a note, “If CMU reveals....... executed correctly”. – Added a second note. *Table 1178: ME_<mode>_MC Field Descriptions*: Removed table footnote in XOSCON field description. *Chapter 58: Debug and Calibration Interface (DCI)* *Section 58.1.3.5: SWT control*: Added note “Debugger should...the SWT.” *Chapter 59: JTAG Controller (JTAGC)* *Table 1233: 6-bit JTAG instructions*: Code instruction “111100” is set to reserved. *Chapter 61: JTAG Data Communication (JDC)* *Figure 1246: Module Status Register (JDC_MSR)*: Corrected register access to read/write.

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	*Chapter 64: Debug Zipwire* *Section 64.8: Debug SIPI module*: Removed "FLEXAY" from this section. *Chapter 65: Debug Serial Interprocessor Interface (SIPI)* *Section 1.6.2: Payload and frame format*: Added Note at the end of the section. *Table 1310: Payload size of LFAST channel frame*: Added footnote. *Figure 13: Data allocation*: Added 32-bit data allocation *Chapter 66: LVDS Fast Asynchronous Serial Transmission – High Speed Debug* *Table 1321: TMCR field descriptions*: Updated description of bitfields CLKTST and LPON. *Table 1325: PICR field descriptions*: updated description of bitfield PNGREQ *Table 1328: UNSTCR field descriptions*: updated description of bitfield USNDRQ *Table 1326: PLLCR field descriptions*: – Corrected typo in FBDIV. – Updated the description of FBDIV. *Chapter 72: Cyclic Redundancy Check (CRC) Unit* *Figure 2: CRC Configuration Register n (CRC_CFGn)*: Swapped reset values versus function of n in the figure footnote of POLYG reset. *Chapter 73: Memory Error Management Unit (MEMU)* *Table 1452: SYS_RAM_UNCERR_ADDR field descriptions*: Removed to "This register is not writable when the corresponding VLD bit is zero." *Table 1457: PERIPH_RAM_UNCERR_ADDR field descriptions* and *Table 1462: FLASH_UNCERR_ADDR field descriptions*: deleted the sentence "This register is not writable when the corresponding VLD bit is zero.)" *Table 1459: FLASH_CERR_STSn register field descriptions*: Updated description of bitfield BAD_BIT to replace 0xFF by 0xFFFF *Chapter 74: Indirect Memory Access (IMA)* *Table 1466: E2EECSR0 field descriptions*: Corrected register name in title. *Chapter 75: Fault Collection and Control Unit (FCCU)* *Table 1486: FCCU_CTRL field descriptions*: Added field numbers *Figure 1532: FCCU Control Register (FCCU_CTRL)*, *Figure 1543: FCCU IO Control Register (FCCU_EINOUT)* and *Figure 1553: FCCU MCS Register (FCCU_MCS)*: Changed some reset bit values to "0/1" with a cross-reference to the Device configuration chapter. *Table 1485: FCCU memory map*: Removed "Register classification" column. *Figure 1537: FCCU RF Status register n (FCCU_RF_Sn)*: Removed "Refer to for the errors description." from table footnote 1. *Table 1504: FCCU_STAT field descriptions*: – Added a note to PhysicErrorPin field description. – Updated this table.

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	*Table 1508: F2A_STATUS field descriptions*: – Changed the table title from F2AF_STATUS to F2A_STATUS – Description of value 0xFF of bitfield FAFS: replaced word “critical” by word “recoverable” *Section 75.6.8: FCCU RF Enable Register n (FCCU_RF_En)*: Deleted text “for debugging purposes”. *Section 75.6.13: FCCU Status Register (FCCU_STAT)*: Deleted text “for debugging purposes”. Deleted text “This register is for test/ debug purposes” from the following sections: – *Section 75.6.14: FCCU NA Freeze Status Register (N2AF_STATUS)* – *Section 75.6.15: FCCU AF Freeze Status Register (A2FF_STATUS)* – *Section 75.6.16: FCCU NF Freeze Status Register (N2FF_STATUS)* – *Section 75.6.17: FCCU FA Freeze Status Register (F2A_STATUS)* *Table 1522: FCCU_EOUT_SIG_ENn field descriptions*: Added a table footnote to enumerated value 0 description in EOUTEN[31:0] field descriptions. *Chapter 76: Self-Test Control Unit (STCU2)* *Section 76.4.2: Reset management*: Added a third sentence. *Section 76.4.8.1: Initialization sequence*: Updated second bullet with a second sentence. *Table 1524: STCU2 memory map*: Updated MBIST registers with k = 0 to 34. *Table 1525: STCU_RUN field descriptions*: Updated MBSPLLEN field description. *Table 1526: STCU_RUNSW field descriptions*: Updated MBSWPLLEN field description. *Table 1528: STCU_CFG field descriptions*: Updated CLK_CFG field description. *Table 1530: STCU_WDG field descriptions*: Updated WDGEOC field descriptions. *Table 1546: STCU_MB_CTRLk field descriptions*: Updated PTR field description. *Figure 1583: STCU2 MBIST Control k Register (STCU_MB_CTRLk)*: Updated offset values with k = 0 to 34. *Section 76.6: Use cases and limitations*: Added paragraph: “The STCU setup has to be defined based on the application-specific safety, boot time and consumption requirements. Contact ST representative for support.”. Removed following register sections: – STCU2 Offline MBIST 3 Status Register (STCU_MBS3) – STCU2 Offline MBIST 3 End Flag Register (STCU_MBE3) – STCU2 Online MBIST 3 Status Register (STCU_MBS3SW) – STCU2 Online MBIST 3 End Flag Register (STCU_MBE3SW) – STCU2 MBIST 3 Unrecoverable FM Register (STCU_MBUFM3) – STCU2 Offline MBIST 4 Status Register (STCU_MBS4) – STCU2 Offline MBIST 4 End Flag Register (STCU_MBE4) – STCU2 Online MBIST 4 Status Register (STCU_MBS4SW) – STCU2 Online MBIST 4 End Flag Register (STCU_MBE4SW) – STCU2 MBIST 4 Unrecoverable FM Register (STCU_MBUFM4)

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	For all available MBIST registers: – Added note, "Refer to..... number of NMCUT available in this device." – Added note to all fields from 0:30, "The availability of this field is based on......Device configuration chapter" *Section 76.4.8.2: MBIST scheduling*: removed STCU_MBE3/4 and STCU_MBE3SW/4SW inside this section. *Section 76.4.9: CRC*: Added note at the end of the section "The CRC calculated depends on the specific sample and not only the self-test configuration programmed." *Section 76.6.1: Offline self-test sequence*: removed STCU_MBS3/4, STCU_MBS3/4 and STCU_MBE3/4 registers inside this section. *Section 76.6.2: Online self-test Sequence*: removed STCU_MBUFM3/4, STCU_MBS3SW/4SW and STCU_MBE3SW/4SW registers inside this section. *Chapter 78: Password and Device Security Module (PASS)* *Section 78.1.2: Features* – Removed bullet "Production Disable control". – Removed section "External signal description". *Section 78.2: Memory map and register definition:* Removed section "Production Disable". *Section 78.2.1.5: Password Group n - Lock 0 Status register (LOCK0_PGn):* Removed paragraph about production disable. *Table 1559: LOCK3_PGn register field descriptions*: Added note to all RLx field descriptions. *Section 78.4.5.2.4: **Fuse Bypass Password compare***: – Removed text "of the Flash page contain different data not related to Fuse Bypass" at the end of the first sentence. – Removed "as the most significant 32 bits" in the second sentence. *Section 78.4.2: Debug Interface Access*: Removed "and production Disable" from the first paragraph. *Table 1563: Debug Interface Enable Truth table*: Removed Production Disable mode row. *Table 1564: Flash Memory Read Protection truth table*: Removed the second table. *Chapter 80: Wakeup unit (WKPU)* *Section 80.1.2: Features*: – Changed "configurable pullup" by "configurable pull". – Added a note referring to the Device Configuration chapter. *Section 80.3.2.5: Wakeup Request Enable Register (WRER)*: Added a note "Any wakeup pad..." *Section 80.3.2.8: Wakeup/Interrupt Filter Enable Register (WIFER)*: Added a Note, "Glitch filter must be enabled in case the wake-up sources are used to exit from Standby." *Section 80.3.2.9: Wakeup/Interrupt Pull Enable Register (WIPUER)*: – Updated the text from, "This register ...... value of 1" to "This register...... SIUL2_MSCR_IO_n.WPDE-WPUE bits."

Table 1602. Document revision history (continued)

Date	Revision	Changes
09-Apr-2020	3 (cont'd)	– *Table 1592: WIPUER field descriptions*: Replaced "Pullup" to "Pull" in the complete table. – Added a table note referring to the Device Configuration chapter on "Description" heading. *Chapter 81: Real Time Clock / Autonomous Periodic Interrupt (RTC/API)* *Section 81.4.6: RTC Compare Value Register (RTC_RTCVAL)*: – Added a note "If RTC interrupt is enabled (RTCIE), it is required after each RTC counter match to reload the RTCVAL via software, to re-enable the interrupt again." – Updated note.
13-Apr-2021	4	Following are the changes in this version of the document: Editorial changes throughout the document Updated the sub-title of the document. *Chapter 2: Introduction* *Figure 3: Periphery allocation*: updated "XBIC_Concentrator_0, 1" to "XBIC_Concentrator_1". *Chapter 3: Embedded memories* *Section 3.2.1: System SRAM*: updated the text from, "The SRAM can be configured....PFCR1.....controller." to "The SRAM can be configured... PRCR1.....controller." *Chapter 6: Functional safety* *Table 14: FCCU failure inputs*: – Updated Failure Description for FCCU channel 17-18 – Updated failure description for FCCU channel 30 – Updated the "Failure description" column of "COMPENSATION CELLS" Updated *Section 6.4.3.1: ECC for storage* *Table 15: ECC RAM implementations*: – updated column-head from memory classification to MEMU classification – added footnote for column-headed ECC for System RAM for location Platform – updated MEMU Classification for location HSM – updated column-headed ECC for Nexus for location Platform *Chapter 7: Device configuration* Updated this chapter. *Chapter 8: Reset and Boot* *Table 97: Module status during reset phases*: – removed note from Module "System watchdog" – updated row-headed SWT

Table 1602. Document revision history (continued)

Date	Revision	Changes
13-Apr-2021	4 (cont'd)	Updated *Section 8.3.5: PHASE2[DEST] Phase: flash initial configuration* *Figure 31: Reset Generation Module Reset Sequence*: added line "self-test bypassed" between PHASE3[DEST] and IDLE[FUNC]. *Section 8.3.2: Power-up phase: power stabilization*: updated the section. *Table 99: SSCM boot record search locations*: updated the location of column for Flash 16 KB. *Table 102: BAF boot record search locations*: updated the location of column for Flash 16 KB *Chapter 9: Device Configuration Format (DCF) Records* *Table 107: DCF client list*: added *2.* *Chapter 10: Power management* *Figure 49: Voltage monitors functionality*: Updated the figure. *Section 10.6.3: Brown-out management*: – added text to note "At any rate there are....." – added reference to the Application note. *Chapter 16: System Integration Unit Lite2 (SIUL2)* Updated *Section 16.2.2.1: MCU ID Register #1 (SIUL2_MIDR1)*: *Chapter 15: Core (z420n3) Description* Added *Section 15.9.2.3: DMEM Control Register 1 (DMEMCTL1)* *Chapter 17: Crossbar switch (XBAR)* *Section 17.3.1: XBAR Priority Registers Slave (XBAR_PRSn)*: – added text "Once the RO.....initiating the write" – added note "No two available.... will not be updated" – removed text "The XBAR_PRSn register is read-only...." *Chapter 18: Crossbar Integrity Checker (XBIC)* *Table 220: XBIC_EIR field descriptions*: In the second note, replaced "SMPU logical bus master assignments" with "AHB Master ID Assignments". *Chapter 25: Clocking* *Section 25.5.1.6: Loss of clock detection*: removed the last sentence "Software may enable or disable an optional backup clock in the PLL". *Table 304: Maximum auxiliary level clock frequencies*: – updated the third note from DSPI_CLK to DSPI_CLK0/1. – added note "DSPI_CLK frequency should always be greater than or equal to 1.25 times PBRIDGE_CLK" to DSPI_CLK1. *Chapter 26: Dual PLL digital interface (PLLDIG)* *Section 26.4.1: Normal mode with reference, PLL0 or both PLLs enabled*: added in Note cross-ref to *Section 26.7: Initialization information*.

Table 1602. Document revision history (continued)

Date	Revision	Changes
13-Apr-2021	4 (cont'd)	Removed section "Register classification for safety requirements" *Section 26.5.2.1: PLL0 control register (PLL0CR)*, *Section 26.5.2.4: PLL1 control register (PLL1CR)*: updated note for field LOLIE. *Chapter 27: Clock Monitor Unit (CMU)* *Section 27.4.1.5: CMU Interrupt Status Register (CMU_ISR)*: updated Note *Chapter 33: Platform RAM controller AHB (PRAMC_AHB)* *Section 33.2.1: Registers description*: added this section title. *Chapter 35: Embedded Flash Memory* *Table 405: UTEST block memory map*: added note to offset 0x140 *Section 35.3.2.4: Lock 0 register (LOCK0)*, *Section 35.3.2.5: Lock 1 register (LOCK1)*, *Section 35.3.2.6: Lock 2 Register (LOCK2)*, *Section 35.3.2.7: Lock 3 Register (LOCK3)*, *Section 35.3.2.8: Alternate Lock 0 register (LOCK0A)*,*Section 35.3.2.9: Alternate Lock 1 register (LOCK1A)*, *Section 35.3.2.10: Select 0 register (SEL0)*, *Section 35.3.2.11: Select 1 register (SEL1)*, *Section 35.3.2.12: Select 2 register (SEL2)*, *Section 35.3.2.13: Select 3 register (SEL3)*: text modified as follow "once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared" in the bit-field description *Table 407: MCR field descriptions*: Description text modification, removed Tpabt /Teabt parameters *Section 35.4.7.1: Array Integrity Self Check* and *Section 35.4.7.2: Margin Read*: Description for enable of Single and double error correction fixed. *Table 410: MCRE field descriptions*: changed coding of N32KL ('10' assigned to One 32KB sector in Low Space) Updated *Figure 362: Erase of blocks (example)* Updated *Figure 365: Array Integrity Check of blocks (example)* *Chapter 36: Flash Memory Programming and Configuration* *Figure 370: Sample DCF record for secure write protection*: updated the value of the figure. *Section 36.1: Selection of Flash memory blocks for erase*: replaced LOCKx with SELx in this section. *Chapter 38: Analog-to-Digital Converters (ADC) Configuration* *Section 38.1: ADC overview*: Updated "four 12-bit SAR ADCs" to "three 12-bit SAR ADCs. *Section 38.2.1: Successive Approximation Register Analog-to-Digital Converter (SAR ADC)*: Updated "four independent 12-bit" to "three independent 12-bit". *Chapter 39: Successive Approximation Register Analog-to-Digital Converter (SARADC)* *Section 39.5.1.11: Watchdog Threshold Register n (WTHRHLRn)*: Replaced in 1st note LSB=0 bits numbers by MSB=0 bit numbers. *Chapter 43: Software Watchdog Timer (SWT)* Removed section "External signal description".

Table 1602. Document revision history (continued)

Date	Revision	Changes
13-Apr-2021	4 (cont'd)	*Chapter 46: CAN subsystem* *Figure 518: CAN subsystem generic block diagram*: updated the "Peripheral Bridge" blocks to "PFBRIDGE". *Chapter 47: Ethernet* *Figure 646: MMC Rx Interrupt mask Register (MMC_RX_INTERRUPT_MASK)*: updated the register access type from User Read to Software read/write. Access type for all the bits ranging from [4:31] changed from Read only to Read Write. *Figure 647: MMC Tx Interrupt Mask Register (MMC_TX_INTERRUPT_MASK)*: updated the register access type from User Read to Software read/write. Access type for all the bits ranging from [4:31] changed from Read only to Read Write. *Chapter 48: Inter-Integrated Circuit (I2C)* Updated *Figure 868: Flow-Chart of DMA mode master receive* *Section 48.4: Memory map and register definition*: added *Section 48.4.2: Registers definition* sub-section and updated registers sections numbering. *Chapter 49: Deserial Serial Peripheral Interface (DSPI)* *Section 49.4.6.3: Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)*: added a second note. Removed section "Register classification for safety" *Section 49.4.12.10: Receive FIFO Drain interrupt or DMA Request*: – added 3 notes. – replaced 'fill' by 'drain' in the first Note. *Chapter 50: LINFlexD* *Section 50.4: Memory map and register description*: added note that refer to the LinFlexD configuration of the Device Configuration chapter. *Table 1024: LINFlexD memory map*: removed note that refer to the Device Configuration chapter and all cross-references to this note. *Table 1049: DMATXE field descriptions*: field editorial changes to remove the table note. *Table 1050: DMARXE field descriptions*: field editorial changes to remove the table note. *Chapter 51: Reset Generation Module (MC_RGM)* Updated *Section 51.1.3: Reset sources* Updated *Section 51.4.2: 'Destructive' resets* *Chapter 54: Power management controller digital interface (PMC_Dig)* Removed all the occurrences of "FlexRay" from the chapter. Updated *Section 54.3: IPS bus interface* *Table 1161: BIST_CTRL field descriptions*: updated description for row-headed NCFEN.

Table 1602. Document revision history (continued)

Date	Revision	Changes
13-Apr-2021	4 (cont'd)	*Chapter 56: Mode Entry Module (MC_ME)* Updated row-headed PDO in *Table 1178: ME_<mode>_MC Field Descriptions* Updated *Section 56.4.2.3: SAFE mode* Updated *Section 56.4.3.16: Pad switch-off* *Chapter 58: Debug and Calibration Interface (DCI)* Updated *Section 58.1.3.5: SWT control* *Chapter 63: JTAG Master (JTAGM)* Updated row-headed IIE in *Table 1298: JTAGM_MCR register field descriptions* Updated *Table 1300: JTAGM_DOR0 register field descriptions* Updated *Table 1301: JTAGM_DOR1 register field descriptions* Updated *Table 1302: JTAGM_DOR2 register field descriptions* Updated *Table 1303: JTAGM_DOR3 register field descriptions* *Chapter 64: Debug Zipwire* *Section 64.8: Debug SIPI module*: removed the text "Flexray (when available)". *Chapter 65: Debug Serial Interprocessor Interface (SIPI)* Updated the chapter *Chapter 66: LVDS Fast Asynchronous Serial Transmission – High Speed Debug* *Figure 1334: LFAST frame structure*: added "(bits)" to "Payload size" and "Frame size" columns headers. *Table 1316: Header payload sizes*: added "(bits)" to "Payload size" and "Frame size" columns headers. *Table 1334: RIISR field descriptions*: updated ICPFF. *Figure 1366: Rx Loopback mode* to *Figure 1371: Tx LVDS Loopback (external) mode with automatic frame generation*: added note regarding CSR. *Chapter 67: Development Trigger Semaphore (DTS)* *Section 67.4.1.1: DTS output enable register (DTS_ENABLE)*: Added DTS_EN_B field. *Chapter 72: Cyclic Redundancy Check (CRC) Unit* *Table 1440: CRC unit memory map*: removed the "Register classification column" *Chapter 74: Indirect Memory Access (IMA)* Updated *Figure 1504: Cache Debug Access Control register (CDACNTL)* Updated *Figure 1505: Cache Debug Access Data register (CDADATA)* Updated *Section 74.2.2.1: e2eECC Error Control/Status register 0 (E2EECSR0)*

Table 1602. Document revision history (continued)

Date	Revision	Changes
13-Apr-2021	4 (cont'd)	*Chapter 75: Fault Collection and Control Unit (FCCU)* *Section 75.6.1: FCCU Control Register (FCCU_CTRL)*: updated the bullet "configure the glitch filter present on location given by FILTER_POSITION parameter" to "configure the glitch filter present on EIN0 input (FCCU failure input #96)" and replaced text "(FCCU_RF_Sn[RFSx] where y = FILTER_POSTN)" to "FCCU failure input #96". Updated *Figure 1516: FCCU block diagram* Updated row-headed "REG intf" for *Table 1476: FCCU submodules* Updated row-headed OPR for *Table 1486: FCCU_CTRL field descriptions* *Chapter 76: Self-Test Control Unit (STCU2)* *Section 76.4.9: CRC*: updated the note "The expected value of the CRC depends on multiple internal critical signals and can differ among different samples." *Table 1530: STCU_WDG field descriptions*: updated description for WDGEOC. *Section 76.5.5: STCU2 PLL Configuration Register (STCU_PLL_CFG)*: added a note, "f_{RC} is the frequency if the internal RC oscillator (that is 16 MHz). f_{PLLin} must be higher than 8 MHz." *Section 76.5.8: STCU2 CRC Expected Status Register (STCU_CRCE)*: added the note to "The expected value of the CRC depends on multiple internal critical signals and can differ among different samples." *Chapter 80: Wakeup unit (WKPU)* *Section 80.3.2.9: Wakeup/Interrupt Pull Enable Register (WIPUER)*: updated the note from "This register shall be left at '0' default value" to "WIPUER bits related to those bonded LP pads that are used to wake-up the device from STANDBY Mode, should be left at 0 default value". *Chapter 81: Real Time Clock / Autonomous Periodic Interrupt (RTC/API)* *Figure 1634: RTC/API block diagram*: Added "32 kHz SXOSC" to the figure.