RM0505
Reference manual

Ultra-low power wireless 32-bit MCU Arm®-based Cortex®-M0+ with Bluetooth® LE and 2.4 GHz radio solution

Introduction

This reference manual provides complete information on how to use the STM32WB09xE microcontroller memory and peripherals.

The STM32WB09xE is a powerful and ultra-low-power 2.4 GHz RF transceiver with an Arm® Cortex®-M0+ microcontroller that can operate up to 64 MHz.

The STM32WB09xE is suitable to implement applications compliant with the Bluetooth® LE SIG specification.

STM32WB09xE microcontrollers include ST state-of-the-art patented technology.

Related documents

The following documents are available from the STMicroelectronics web site, www.st.com:

• STM32WB09xE datasheet (DS14210)

• STM32WB09xE errata sheet (ES0584)

For information on the Cortex®-M0+ core, refer to the corresponding Technical Reference Manuals, available from the www.arm.com website.

For information on Bluetooth® refer to www.bluetooth.com.

Contents

List of tables

List of figures

1 Documentation conventions

1.1 General information

For information on the Arm[a] Cortex®-M0+ core, refer to the Cortex®-M0+ Technical Reference Manual, available from the http://www.arm.com website.

For information on Bluetooth® refer to the www.bluetooth.com website.

1.2 List of abbreviations for registers

The following abbreviations are used in register descriptions:

read/write (rw or R/W)	Software can read and write to these bits.
read-only (r or R)	Software can only read these bits.
write-only (w or W)	Software can only write to this bit. Reading the bit returns the reset value.
read/write-once (RWOnce)	Software can read these bits but write is only allowed once.
read/clear (rc_w1 or RWC1)	Software can read as well as clear this bit by writing 1. Writing ‘0’ has no effect on the bit value.
read/clear (rc_w0 or RWC0)	Software can read as well as clear this bit by writing 0. Writing ‘1’ has no effect on the bit value.
read/clear by read (rc_r or RC)	Software can read this bit. Reading this bit automatically clears it to ‘0’. Writing ‘0’ has no effect on the bit value.
read/set (rs or RWS1)	Software can read as well as set this bit. Writing ‘0’ has no effect on the bit value.
read-only write trigger (rt_w or RWH)	Software can read this bit. Writing ‘0’ or ‘1’ triggers an event but has no effect on the bit value.
toggle (t or RWT1)	Software can only toggle this bit by writing ‘1’. Writing ‘0’ has no effect.
Reserved (Res.)	Reserved bit, must be kept at reset value.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.3 Glossary

This section gives a brief definition abbreviations used in this document:

- The SoC integrates one debug port sharing same pins:
  - SWD debug port (SWD-DP) provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol.
- Word: data/instruction of 32-bit length.
- Half word: data/instruction of 16-bit length.
- Byte: data of 8-bit length.
- Double word: data of 64-bit length.
- AHB: advanced high-performance bus.
- APB: advanced peripheral bus.
- CPU: refers to the Cortex®-M0+ core.

1.4 Availability of peripherals

For availability of peripherals and their number across all sales types, refer to the particular device datasheet.

1.5 Acronyms

Table 1. Acronyms

Acronym	Description
ADC	Analog to digital converter
AES	Advanced encryption standard hardware accelerator
AGC	Automatic gain converter
AHB	Advanced high-performance bus
APB	Advanced peripheral bus
AoA (or AOA)	Angle of arrival
AoD (or AOD)	Angle of departure
BOR	Brown out reset
BPU	Breakpoint unit (Arm debug component)
CPU	Central processor unit
CRC	Cyclic redundancy check
CTE	Constant tone extension
CRS	Clock Recovery System
CSS	Clock Security System
DBG	Debug
DMA	Direct memory access
DMAMUX	Direct memory access multiplexer

Table 1. Acronyms

Acronym	Description
DWT	Data watchpoint and trace (Arm debug component)
FSM	Finite state machine
ECC	Error Code Correction
ESE	Enable Security Environment
ETM	Embedded Trace Module
EXTI	External Interrupts and Event controller
FPB	Flash Patch and Breakpoint unit (ARM debug component)
FPU	Floating Point Unit
GPIO	Input output
HSE	High speed external clock oscillator
HSEM	Hardware SEMaphore
HSI	High speed internal clock oscillator
HW	Hardware
I2C	Inter integrated circuit (communication standard)
I2S	Inter integrated (communication standard)
IAP	In Application Programming
ICP	In Circuit Programming
IPCC	Inter Processor Communication Controller
IRQ	Interrupt request
ITM	Instrumentation trace macrocell (Arm debug component)
IWDG	Independent watchdog
LDO	Low drop output
LP	Low power
LPM	Low power manager
LSB	Least significant byte
LSE	Low speed external clock oscillator
LSI	Low speed internal clock oscillator
MCU	Microcontroller unit
MPU	Memory protection unit
MR_BLE	Radio subsystem
MSB	Most significant byte
NVIC	Nested vector interrupt controller
OBL	Option byte loading
OSC	Oscillator
OTP	One time programmable

Table 1. Acronyms

Acronym	Description
PLL	Phase locked loop
PA	Power amplifier
POR	Power on reset
PVD	Programmable voltage detector
PVM	Peripheral voltage monitoring
PWR	Power controller
RC	Resistor capacitor oscillator
RCC	Reset and clock controller
RDP	ReaD Protection
RF	Radio frequency
RF2G4	Analog radio block used with the radio controller MR_BLE IP.
RFU	Reserved for future use
RRM	Radio resource manager
Rx	Reception
SoC	System on chip. The device embedding the radio controller MR_BLE IP described in this document.
ROM	Read only memory
RSS	Root Security Services (Cortex-M0+ secure boot FW)
RSSI	Receive Signal Strength Indication
RTC	Real time clock
SMPS	Switch mode power supply
SPI	Serial peripheral interface (communication standard)
SRAM	Static random access memory
SW	Software
Tx	Transmission
UDRA	Unified direct register access (part of the RRM block)
SWD	Single Wire debug
SWJ-DP	Single Wire Joint Test Access Group - Debug Port (Arm debug component)
SYSCFG	System Configuration
TIM	Timer
TPIU	Trace Port Interface Unit (ARM debug component)
TSC	Touch Sensing Controller
USART	Universal synchronous asynchronous receiver transmitter (communication standard)
Vbat	Battery voltage. Voltage used for the always-on part of the design.

Table 1. Acronyms

Acronym	Description
UID	Unique IDentification number
USB-FS	Universal Serial Bus - Full Speed (communication standard)
VCO	Voltage Controlled Oscillator
VREF	Voltage reference
WFE	Wait For Event (ARM instruction entering low power mode)
WFI	Wait for instruction (Arm instruction entering low power mode)
WKUP	Wakeup
WRP	Write protection
WDG	Watchdog

1.6 Reference documents

Table 2. Document references

Reference number	Name	Owner	Revision
[1]	STM32WB09xE datasheet, DS14210	STMicroelectronics	Latest revision
[2]	Bluetooth Core Specification	Bluetooth SIG	5.4
[3]	Cortex®-M0+ Technical Reference Manual	Arm Ltd. http://www.arm.com	Latest version
[4]	ARMv6-M Architecture Reference Manual	Arm Ltd. http://www.arm.com	Latest version

2 Memory and bus architecture

2.1 System architecture

The main system consists of 32-bit multilayer AHB bus matrix that interconnects:

- Three masters:
  - CPU (Cortex®-M0+) core S-bus
  - DMA
  - Radio system
- Seven slaves:
  - Internal flash memory on CPU (Cortex®-M0+) S bus
  - Internal SRAM0 (16 Kbytes)
  - Internal SRAM1 (16 Kbytes)
  - Internal SRAM2 (16 Kbytes)
  - Internal SRAM3 (16 Kbytes)
  - APB0 peripherals (through an AHB to APB bridge)
  - APB1 peripherals (through an AHB to APB bridge)
  - AHB0 peripherals
  - AHBRF including AHB to APB bridge and Radio peripherals (connected to APB2)

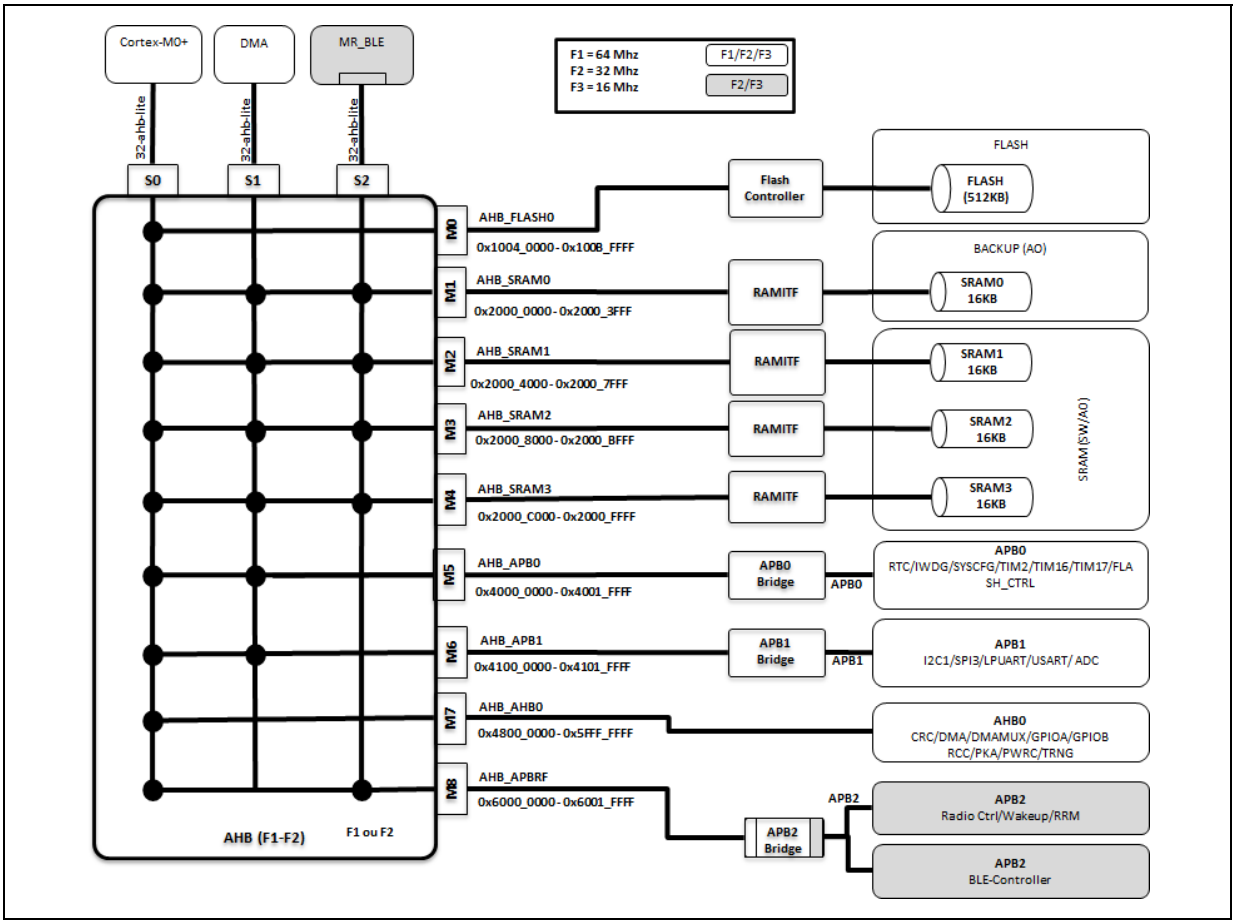
The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in *Figure 1: System architecture*:

The system consists of a Cortex®-M0+ “Radio protocol and application” processor with its radio sub-system.

There is a single flash memory to be used by the CPU for both Bluetooth LE protocol and application management.

The peripherals are located on the different system buses (AHB, APB0, APB1, APB2 for the radio system). There are four SRAM banks: a SRAM0 always power supplied and SRAM1,2,3 that can be programmed to be always on or switchable. For more information see *Section 5.7.2: Control register 2 (PWRC_CR2)*.

Figure 1. System architecture



2.1.1 S0: CPU (Cortex®-M0+) S-bus

This bus connects the system bus of the CPU core to the bus matrix. This bus is used by the core to fetch instructions, for literal load and debug access, and access data located in a peripheral or SRAM area. The targets of this bus are all the possible peripherals (the internal flash and SRAM memories, the AHB0, APB0, APB1 and APB2 peripherals).

2.1.2 S1: DMA-bus

This bus connects the AHB master interface of the DMA to the bus matrix.The targets of this bus are the two banks of SRAM, the APB0 and APB1 peripheral.

2.1.3 S2: Radio system-bus

This bus connects the AHB master interface of the Radio system to the bus matrix.The targets of this bus are the two banks of SRAM and the APB2 peripherals (internal APB blocks of the MR_BLE IP).

2.1.4 Bus matrix

The bus matrix manages the access arbitration between masters. The arbitration uses a Round Robin algorithm. The bus matrix is composed of three masters (CPU, DMA1-bus and Radio system-bus) and seven slaves (FLASH, SRAM0, SRAM1,SRAM2, SRAM3, APB0 and APB1, AHB0 and AHBRF).

AHB/APB bridges

The two bridges AHB to APB0 and AHB to ABP1 provide full synchronous connections between the AHB and the two APB buses.

The bridge AHB to APB2 provides synchronous connections between the AHB and the APB bus. Two blocks are added on the AHB/APB path to manage potential prescaled MR_BLE frequency versus the system frequency:

- AHB Up Converter is used for the MR_BLE AHB master transactions towards the SRAM.
- AHB Down Converter is used for the CPU AHB master transactions towards the MR_BLE APB registers.

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to this bridge.

After each device reset, all peripheral clocks are disabled (except for the SRAM and flash memory interface). Before using a peripheral you have to enable its clock in the RCC_AHBxENR and the RCC_APBxENR registers.

Note: When a 16- or 8-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 16- or 8-bit data to feed the 32-bit vector.

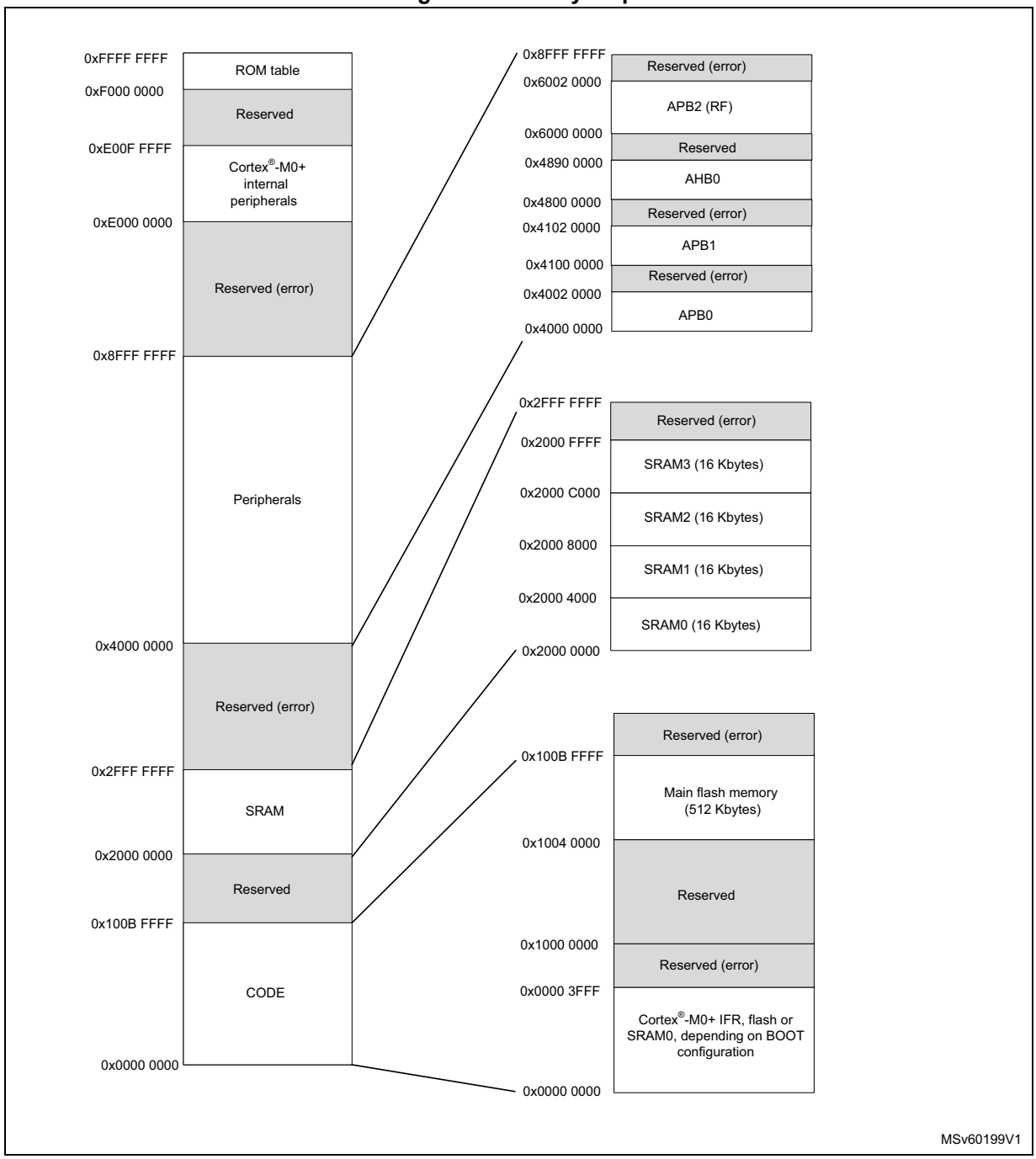
2.2 Memory organization

2.2.1 Introduction

Program memory, data memory and registers are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in Little Endian format. the lowest numbered byte in a word is considered the words least significant byte and the highest numbered byte the most significant.

Figure 2. Memory map



All the memory areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory and registers areas, refer to *Section 2.2.2: Memory map and register boundary addresses*.

2.2.2 Memory map and register boundary addresses

Table 3 gives the boundary addresses of the peripherals available in the device.

Table 3. Memory map and peripheral register boundary addresses

Bus	Boundary address	Actual size (bytes)	Peripheral	Peripheral register map
Misc	0xE00F FFFF - 0xFFFF FFFF	255 M	Reserved	-
	0xE000 0000 - 0xE00F FFFF	1 M	Private peripheral bus	Cortex-M0+ registers (interrupt controller, SysTick, etc.)
APB2	0x6000 1800 - 0x6000 1BFF	1 K	WAKEUP	Wakeup registers of the MR_BLE.
	0x6000 1500 - 0x6000 17FF	1 K	Radio Registers	Radio registers through APB direct access.
	0x6000 1400 - 0x6000 14FF		RRM	RRM registers of the MR_BLE.
	0x6000 1000 - 0x6000 13FF	1 K	RADIO_CTRL	Radio controller registers of the MR_BLE.
	0x6000 0000 - 0x6000 00FF	256	Bluetooth LE	Bluetooth LE link layer registers of the MR_BLE.
AHB0	0x4880 0000 - 0x4880 03FF	1 K	DMAMUX	See *Section 11.5.2: DMAMUX register map*
	0x4870 0000 - 0x4870 00FF	256	DMA slave	See *Section 10.4.7: DMA register map*
	0x4860 0000 - 0x4860 0FFC	1 K	TRNG	See *Section 14.7: TRNG registers*
	0x4850 0000 - 0x4850 03FF	1 K	PWRC	See *Section 5.7.22: PWRC register map*
	0x4840 0000 - 0x4840 03FF	1 K	RCC	See section *Section 6.6.18: RCC register map*
	0x4830 0400 - 0x4830 23FF	8 K	PKA RAM	See *Section 13.6.5: PKA register map* for details
	0x4830 0000 - 0x4830 03FF	1 K	PKA slave	
	0x4820 0000 - 0x4820 03FF	1 K	CRC	See *Section 15.4.6: CRC register map*
	0x4810 0000 - 0x4810 03FF	1 K	GPIOB	See section *Section 7.4.23: GPIO register map*
	0x4800 0000 - 0x4800 03FF	1 K	GPIOA	See section *Section 7.4.23: GPIO register map*

Table 3. Memory map and peripheral register boundary addresses

Bus	Boundary address	Actual size (bytes)	Peripheral	Peripheral register map
APB1	0x4100 7000 - 0x4100 73FF	1 K	SPI3	See *Section 24.9.10: SPI/I2S register map*
	0x4100 6000 - 0x4100 60FF	256	ADC	See *Section 12.6.18: ADC register map*
	0x4100 5000 - 0x4100 53FF	1 K	LPUART	See *Section 23.5.11: LPUART register map* for more details
	0x4100 4000 - 0x4100 43FF	1 K	USART	See *Section 24.9.10: SPI/I2S register map*
	0x4100 3000 - 0x4100 33FF	1 K	Reserved	-
	0x4100 2000 - 0x4100 23FF	1 K	Reserved	-
	0x4100 1000 - 0x4100 13FF	1 K	Reserved	-
	0x4100 0000 - 0x4100 03FF	1 K	I2C1	See *Section 21: Inter-integrated circuit (I2C) interface*
APB0	0x4000 6000 - 0x4000 63FF	1 K	TIM17	See *Section 17.4.19: TIM16/TIM17 register map*
	0x4000 5000 - 0x4000 53FF	1 K	TIM16	
	0x4000 4000 - 0x4000 43FF	1 K	RTC	See *Section 19.6.14: RTC register map*
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	See *Section 20.4.6: IWDG register map*
	0x4000 2000 - 0x4000 23FF	1 K	TIM2	See *Section 17.4.19: TIM16/TIM17 register map*
	0x4000 1000 - 0x4000 1FFF	4 K	FLASH_CTRL	See *Section 9.3.11: Flash controller register map*
	0x4000 0000 - 0x4000 003F	64	SYSTEM_CTRL	See *Section 8.2.16: System controller register map*
SRAM	0x2000 C000 - 0x2000 FFFF	16 K	SRAM3	-
	0x2000 8000 - 0x2000 BFFF	16 K	SRAM2	-
	0x2000 4000 - 0x2000 7FFF	16 K	SRAM1	-
	0x2000 0000 - 0x2000 3FFF	16 K	SRAM0	-
	0x2000 0018-0x2000 3FFF	16360		Start of user RAM
	0x2000 0000-0x2000 0017	24		Reserved. See *Table 4: SRAM0 reserved locations*.
Flash	0x1004 0000 - 0x100B FFFF	512 K	Main flash	-
OTP	0x1000 1800 - 0x1000 1BFF	1 K	OTP area	-
Boot(1)	0x0000 0000 - 0x0000 3FFF	16 K	CPU Boot area	Mirroring of flash or SRAM0

1. This area is a mirroring area. The CPU accesses are redirected to other memory map depending in REMAP bits located in the flash controller CONFIG register. See *Table 5* for remapping detail.

Table 4. SRAM0 reserved locations

Address	Actual size (bytes)	Identifier	Description
0x2000 0000	4	Reserved1	Reserved for future use
0x2000 0004	4	Reserved2	Reserved for future use
0x2000 0008	4	SavedMSP	Used by the LPM module. Used to save context information pointer.
0x2000 000C	4	WakeupFromSleepFlag	Used by the LPM module. Indicates whether the system has been woken up from Deepstop mode.
0x2000 0010	4	ResetReason	Copy of last reset reason from register RCC_CSR. The register is read, copied to this location by the bootloader code and finally cleared. As a consequence, software reading the register always reads 0. Users should read this location to know the last reset reason.
0x2000 0014	4	AppBase	Relocation of application base. The Bootloader jumps to the location pointed to by this value when a wakeup from DeepStop occurs.
0x2000 00C0	CFG_NUM_RADIO_TASKS*92+28	Blue Core Config	Only when radio is used and radio clock is activated. Note: CFG_NUM_RADIO_TASKS is the number of simultaneous radio tasks selected by the application (radio controller supports up to 128 simultaneous radio tasks, but actual usable max value depends on the available RAM).

SRAM0 memory locations from 0x2000 0000 to 0x2000 0017 are reserved for system use and users are not allowed to use these locations for their application.

Table 5. Address remapping depending on the REMAP bit

REMAP	Memory mapped
0	Main flash memory
1	SRAM0

2.3 Arm Cortex-M0+

The Arm Cortex-M0+ processor was developed to provide a low-cost platform that meets the needs of MCU implementation, with a reduced pin count and low-power consumption, while delivering outstanding computational performance and an advanced response to interrupts.

The embedded Cortex-M0+ embeds:

- four breakpoints,
- two watchpoints,
- a MPU (Memory Protection Unit) providing eight unified protection regions
- a SysTick running only with the system clock (external clock option not supported).

2.3.1 CPU memory remap

Following CPU boot the application software can modify the memory map at address 0x0000 0000. This modification is performed by programming the REMAP bit in the flash controller (see *Section 9.3.2: Configuration register (CONFIG)*).

The following memory can be remapped:

- Main flash memory
- SRAM0 memory

Embedded boot loader

ST provides a boot loader executed after each CPU reboot. This boot loader has its own documentation.

Note: The STM32WB09xE *device latches the PA8 / PA9 / PA10 / PA11 / PB12 / PB13 / PB14 / PB15 pads value at POR. The information is available in PWRC_SR2 register (see Section 5.7.6: Status register 2 (PWRC_SR2)). One of those eight I/Os can be used by the boot loader as boot indication between a normal boot or a boot on serial interface.*

2.3.2 Interrupts

Interrupts are handled by the Cortex-M0+ nested vector interrupt controller (NVIC). The NVIC controls specific Cortex-M0+ interrupts (address 0x00 to 0x3C) as well as 32 user interrupts (address 0x40 to 0xBC). In the STM32WB09xE device, the user interrupts have been connected to the interrupt signals of the different peripherals (GPIO, flash controller, timer, UART, and so on). These interrupts can be controlled using the ISER, ICER, ISPR and ICPR registers (see the Cortex-M0+ Devices Generic User Guide *[3]*).

Vector table

On reset, the Cortex-M0+ vector table is fixed at address 0x0000_0000. The software may relocate the vector table address to a different memory location, in a range 0x0000_0000 to 0xFFFF_FF80 in multiples of 256 bytes through the Vector Table Offset Register (VTOR) located in the Cortex-M0+ registers area (see the Cortex-M0+ Devices Generic User Guide *[3]*).

The interrupt mapping for the vector table of the STM32WB09xE device is described in *Table 6*.

Table 6. Interrupt vectors (1)

Position	Priority	Type of priority	Acronym	Description	Address offset
-	-	-	-	Initial main SP	0x0000 0000
-	-3	Fixed	Reset	Reset handler	0x0000 0004
-	-2	Fixed	NmiSR	NMI handler	0x0000 0008
-	-1	Fixed	FaultISR	HardFault handler	0x0000 000C
-	-	RESERVED	RESERVED	RESERVED	0x0000 000C - 0x0000 0038
-	6	Settable	SysTick	System tick timer	0x0000 003C
0	Init 0	Settable	NVM	Non-volatile memory (flash) controller	0x0000 0040
1	Init 0	Settable	RCC	Reset and Clock Controller	0x0000 0044
2	Init 0	Settable	BATTERY	PVD	0x0000 0048
3	Init 0	Settable	I2C1	I2C1 interrupt	0x0000 004C
4	Init 0	Settable	RESERVED	RESERVED	0x0000 0050
5	Init 0	Settable	RESERVED	RESERVED	0x0000 0054
6	Init 0	Settable	RESERVED	RESERVED	0x0000 0058
7	Init 0	Settable	SPI3	SPI3 interrupt	0x0000 005C
8	Init 0	Settable	USART	USART interrupt	0x0000 0060
9	Init 0	Settable	LPUART	LPUART interrupt	0x0000 0064
10	Init 0	Settable	TIM2	TIM2 interrupt	0x0000 0068
11	Init 0	Settable	RTC	RTC interrupt	0x0000 006C
12	Init 0	Settable	RESERVED	RESERVED	0x0000 0070
12	Init 0	Settable	ADC	ADC interrupt	0x0000 0070
13	Init 0	Settable	PKA	PKA interrupt	0x0000 0074
14	Init 0	Settable	RESERVED	RESERVED	0x0000 0078
15	Init 0	Settable	GPIOA	GPIOA interrupt	0x0000 007C
16	Init 0	Settable	GPIOB	GPIOB interrupt	0x0000 0080
17	Init 0	Settable	DMA	DMA interrupt	0x0000 0084
18	Init 0	Settable	BLE_TXRX	Bluetooth LE end of transfer interrupt	0x0000 0088
19	Init 0	Settable	RESERVED	RESERVED	0x0000 008C
20	Init 0	Settable	RESERVED	RESERVED	0x0000 0090
21	Init 0	Settable	RADIO_CTRL	Radio control slow clock measurement interrupt	0x0000 0094
22	Init 0	Settable	MR_BLE	RRM and Radio FSM interrupt	0x0000 0098
23	Init 0	Settable	CPU_WKUP	CPU Wakeup interrupt	0x0000 009C
24	Init 0	Settable	BLE_WKUP	Bluetooth LE Wakeup interrupt	0x0000 00A0

Table 6. Interrupt vectors (1)

Position	Priority	Type of priority	Acronym	Description	Address offset
25	Init 0	Settable	BLE_SEQ	Bluetooth LE RX/TX sequence interrupt	0x0000 00A4
26	Init 0	Settable	TIM16	TIM16 interrupt	0x0000 00A8
27	Init 0	Settable	TIM17	TIM17 interrupt	0x0000 00AC
28	Init 0	Settable	TRNG	TRNG interrupt	0x0000 00B0
29 - 31	Init 0	Settable	RESERVED	RESERVED	0x0000 00B4 - 0x0000 00BC

1. Priority level is set to 0 after reset, each interrupt can be programmed with 4 higher priorities.

Interrupt activation sequence

Safely activating a peripheral interrupt requires the following steps:

- make sure the interrupt is disabled and cleared on the peripheral side (this prevents receiving an interrupt due to a previous event)
- clear pending requests for this interrupt on the Cortex-M0+ side using the NVIC ICPR register;
- set the priority using the NVIC IPR0-IPR7 registers;
- activate on the Cortex-M0+ side using the NVIC ISER register;
- activate on the peripheral side.

Note that most peripherals require clearing interrupt requests on the peripheral side when handling interrupt service requests to prevent triggering continuous interrupts for the same event.

For more details on the Cortex-M0+ exception model, see the ARMv6-M Architecture Reference Manual *[4]*, §B1.5.

For more details on the Cortex-M0+ NVIC behavior and registers, see the ARMv6-M Architecture Reference Manual *[4]*, §B3.4.

3 AHB up/down converter

The STM32WB09xE device can support several system clock frequencies from 1 MHz to 32 MHz.

The Bluetooth LE IP does not need more than 32 MHz to achieve the processing of the radio transfers while the system (CPU, DMA, memories) may require higher performance for application purpose.

To avoid useless overconsumption, AHB up/down converter block has been added to introduce an adjustable divider by one, two or four on AHB and APB bus of the MR_BLE (linked to AHBRF / APB2 bridge) versus the system bus matrix frequency. This block allows dividing by one, two or four the system clock for the MR_BLE IP of the device.

When the system and the MR_BLE share the same frequency, the AHB up/down converter block only transfers the AHB signals from one clock domain to the other.

Note: The system clock must be at 16/32 MHz and always equal or faster than MR_BLE clock when radio is used (no other frequencies).

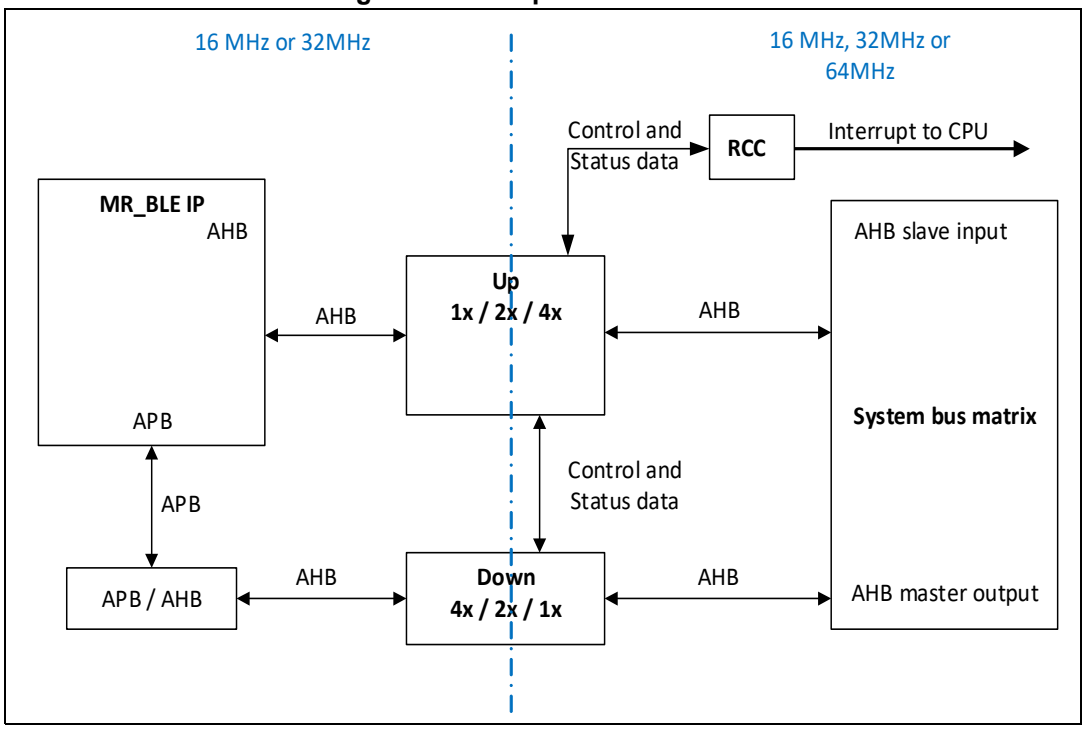
3.1 AHB up/down converter description

The AHB up/down converter role is to allow STM32WB09xE device to support a fast system clock (up to 32 MHz).

The AHBUPCONV block manages:

- proper on-the-fly (up-to-down and down-to-up) frequency switching by safely updating the ratio (one, two, four) between the system and the MR_BLE IP frequencies.
- AHB and APB data transfer between MR_BLE running at the same frequency or half or quarter of the frequency of the rest of the system (AHB and APB) that may run up to 32 MHz.

Figure 3. AHB up/down converter



The management of data transfer versus clock domain and possible clock switch request is done using state machines:

– one for the AHB master up converter
– one for the AHB slave down converter.

When a CPU/system clock frequency switch is needed (activate or deactivate the divider by two between the system and the MR_BLE), the user must request the new system clock targeted frequency in the RCC_CSCMDR.REQUEST bit (see *Section 6.6.6: Clock switch command register (RCC_CSCMDR)* for details).

When receiving a new divider ratio to apply (from the RCC), the AHBUPCONV block:

- Informs the AHBDOWNCONV block (managing the APB transfer from the CPU to the MR_BLE APB registers)
- Check the traffic on the AHB bus between MR_BLE and CPU:
  – if no transfer is on going, it uses the HREADY signal on the bus to hold any potential transaction that could occur during the clock frequency switch.

Note: To respect the AHB lite protocol, the HREADY signal is fallen down only after the address phase of a new transfer, the new transfer phase data being stored internally in the converter.

  - if an AHB transfer is on-going, it wait until the current AHB transfer ends and then hold the AHB traffic as explain above.
- In parallel, the AHBDOWNCONV block does the same action to hold any new transfer on the slave path of the data path (CPU access to MR_BLE APB registers)
- Once the AHBUPCONV block has hold its AHB line and has the confirmation from the AHBDOWNCONV the other AHB2APB data path is also safely hold, it allows the RCC block to change the system clock frequency to the requested one.
- When the new system clock frequency is stable, the RCC informs the AHBUPCONV it is done. Then:
  - the AHBUPCONV release the AHB transfers
  - the AHBUPCONV informs the AHBDOWNCONV is can also releases the AHB/APB transfers.

4 I/O operating modes

The STM32WB09xE device proposes up to 20 programmable I/Os (in all the package options).

Each I/O can be programmed in several modes:

- Input mode
- Output mode
- Alternate function
- Analog mode (when available see *Table 9*)

The STM32WB09xE device supports six alternate modes called AFx (x = 0 to 4 and 6). The configuration of each I/O for those alternate modes is detailed in *Table 7* and *Table 8*.

Note: The I/O features presented in Table 7 and Table 8 are valid only when the associated I/O is programmed as alternate function.

Refer to Section 7: General-purpose I/Os (GPIO) for more details on all configurations.

The number of I/Os available in the STM32WB09xE device depends on the package:

- 20 I/Os in VFQFPN32
- 20 I/Os in WLCSP36

Type acronyms correspond to:

- I: Input
- O: Output
- I/O: Input Output
- OD: Open Drain

Table 7. Alternate modes 0, 1 and 2

Pin name	AF0 mode		AF1 mode		AF2 mode	
	Type	Signal	Type	Signal	Type	Signal
Port A						
PA0(1)	I/OD	I2C1_SCL	I	USART_CTS	O	IR_OUT
PA1(1)	I/OD	I2C1_SDA	O	IR_OUT	I/O	USART_TX
PA2	I/O	DEBUG_SWDIO	I/O	USART_CK	-	-
PA3	I	DEBUG_SWCLK	O	USART_RTS_DE	-	-
PA8	I/O	USART_RX	O	RTC_OUT	O	RADIO_RX_SEQUENCE
PA9	I/O	USART_TX	-	-	O	RTC_OUT
PA10	-	-	I	LPUART_CTS	O	RADIO_TX_SEQUENCE
PA11	O	RCC_MCO	-	-	O	RADIO_RX_SEQUENCE
Port B						
PB0	I/O	USART_RX	O	LPUART_RTS_DE	I/O	TIM16_CH1

Table 7. Alternate modes 0, 1 and 2 (continued)

Pin name	AF0 mode		AF1 mode		AF2 mode	
	Type	Signal	Type	Signal	Type	Signal
PB1	I/O	USART_CK	-	-	O	TIM16_CH1N
PB2	O	USART_RTS_DE	-	-	I/O	TIM16_BRK
PB3	I	USART_CTS	I/O	LPUART_TX	I/O	TIM17_CH1
PB4	I/O	LPUART_TX	-	-	O	TIM17_CH1N
PB5	I/O	LPUART_RX	-	-	I/O	TIM17_BRK
PB6(1)	I/OD	I2C1_SCL	-	-	I/O	TIM17_CH1
PB7(1)	I/OD	I2C1_SDA	-	-	I	USART_CTS
PB12(2)	-	-	O	RCC_LCO	I	LPUART_CTS
PB13(3)	-	-	-	-	-	-
PB14	I/O	I2C1_SMBA	O	RADIO_TX_SEQUENCE	I	TIM2_ETR
PB15	-	-	-	-	-	-

1. This I/O is able to be configured in open-drain.
2. This I/O is shared with OSC32_OUT (low speed clock oscillator pin).
3. This I/O is shared with OSC32_IN (low speed clock oscillator pin).

Table 8. Alternate modes 3, 4, and 6

Pin name	AF3 mode		AF4 mode		AF6 mode	
	Type	Signal	Type	Signal	Type	Signal
Port A						
PA0(1)	-	-	I/O	TIM2_CH3	-	-
PA1(1)	-	-	I/O	TIM2_CH4	-	-
PA2	O	I2S3_MCK	I/O	TIM2_CH1	I/O	TIM16_CH1
PA3	I/O	SPI3_SCK/ I2S3_SCK	I/O	TIM2_CH2	O	TIM16_CH1N
PA8	I/O	SPI3_MISO	I/O	TIM2_CH3	I/O	TIM16_BRK
PA9	I/O	SPI3_NSS/ I2S3_WS	I/O	TIM2_CH4	I/O	TIM17_CH1
PA10	O	I2S3_MCK	I/O	DEBUG_SWDIO	O	TIM17_CH1N
PA11	I/O	SPI3_MOSI/ I2S3_SD	I	DEBUG_SWCLK	I/O	TIM17_BRK
Port B						
PB0	-	-	-	-	O	ANT_ID[0]
PB1	I	TIM2_ETR	-	-	O	ANT_ID[1]
PB2	I/O	TIM2_CH3	-	-	O	ANT_ID[2]
PB3	I/O	TIM2_CH4	I/O	SPI3_SCK/ I2S3_SCK	O	ANT_ID[3]
PB4	-	-	I/O	TIM2_CH1	O	ANT_ID[4]

Table 8. Alternate modes 3, 4, and 6

Pin name	AF3 mode		AF4 mode		AF6 mode	
	Type	Signal	Type	Signal	Type	Signal
PB5	-	-	I/O	TIM2_CH2	O	ANT_ID[5]
PB6(1)	I/O	LPUART_TX	I/O	TIM2_CH1	O	ANT_ID[6]
PB7(1)	I/O	LPUART_RX	I/O	TIM2_CH2	O	RADIO_RF_ACTIVITY
PB12(2)	-	-	I/O	TIM2_CH3	-	-
PB13(3)	-	-	I/O	TIM2_CH4	-	-
PB14	O	RCC_MCO	-	-	I/O	USART_RX
PB15	-	-	-	-	I/O	USART_TX

1. This I/O is able to be configured in open-drain.
2. This I/O is shared with OSC32_OUT (low speed clock oscillator pin).
3. This I/O is shared with OSC32_IN (low speed clock oscillator pin).

Concerning the ADC block present in the STM32WB09xE device:

- The eight ADC channels are available on PA2, PA3, PB0, PB1, PB2, PB3, PB4, PB5

Note: The ADC features on those pins are available if the associated pin is configured in analog mode (see Section 7: General-purpose I/Os (GPIO) for more details).

Table 9 shows the mapping associated to analog configuration (for pins able to support analog mode).

Table 9. I/O Analog feature mapping (1)

Pin name	Analog feature	Parallel Analog feature	Pin name	Analog feature	Parallel Analog feature
Port A			Port B		
PA0	N/A	-	PB0	ADC_VINM1	N/A
PA1	N/A	-	PB1	ADC_VINP1	N/A
PA2	ADC_VINM2	-	PB2	ADC_VINM0	N/A
PA3	ADC_VINP2		PB3	ADC_VINP0	N/A
PA8	N/A	-	PB4	ADC_VINM3	N/A
PA9	N/A	-	PB5	ADC_VINP3	N/A
PA10	N/A	-	PB6	N/A	N/A
PA11	N/A	-	PB7	N/A	N/A
-	-	-	PB12	-	RCC_OSC32_OUT (2)
-	-	-	PB13	-	RCC_OSC32_IN (2)
-	-	-	PB14	-	PVD VIN
-	-	-	PB15	-	N/A

1. N/A means Not Applicable as the associated I/O does not support analog option.

2. The parallel analog feature is obtained by setting the RCC_CR.LSEON bit in the RCC registers. Then the PB12 and PB13 are forced by hardware to manage the LSE through RCC_OSC32_OUT/RCC_OSC32_IN whatever the selected mode in the associated GPIO_MODERx register, but if PWRC_CR1.APC bit is set, it's necessary to disable PUB12, PUB13, PDB12, PDB13 in PWRC registers.

Table 10. I/O Additional function mapping(1)

Pin name	Function	Pin name	Function
Port A		**Port B**	
PA0	Wakeup	PB0	Wakeup
PA1	Wakeup	PB1	Wakeup
PA2	Wakeup	PB2	Wakeup
PA3	Wakeup	PB3	Wakeup
PA8	Wakeup	PB4	Wakeup
PA9	Wakeup	PB5	Wakeup
PA10	Wakeup, RCC_LCO	PB6	Wakeup
PA11	Wakeup	PB7	Wakeup
-	-	PB12	Wakeup, RCC_OSC32_OUT (2)
-	-	PB13	Wakeup, RCC_OSC32_IN (2)
-	-	PB14	Wakeup
-	-	PB15	Wakeup

1. For the additional functions like Wakeup I/O, LSE oscillator, LCO configure the required function in the related PWRC, RCC, RTC registers. These functions have priority over the configuration in the standard GPIO registers.
2. The additional functions for LSE oscillator are obtained by setting the RCC_CR.LSEON bit in the RCC registers. Then the PB12 and PB13 are forced by hardware to manage the LSE through RCC_OSC32_OUT/RCC_OSC32_IN whatever the selected mode in the associated GPIOx_MODER register, but if PWRC_CR1.APC bit is set, it's necessary to disable PUB12, PUB13, PDB12, PDB13 in PWRC registers.

5 Power controller (PWRC)

The Power controller block control the analog supplies block and manages the startup, active and low power phase of the device including the transition from one state to another.

5.1 Features

The Power controller block supports the following features:

- Low power mode choice and entry/exit sequences
- The flash memory power (ON/OFF) and the power down sequence
- RAM banks retention control
- Power monitoring:
  - POR/PDR reset on rising/falling VDDIO voltage
  - Programmable Voltage Detector (PVD) monitoring of the VDDIO with programmable threshold or of an external analog input voltage (compared to the internal VBGP)
- I/Os pull-up/-down during low power mode
- Wakeup I/O configuration

5.2 Power supply domains

The STM32WB09xE device embeds three power domains:

- VDD33 (also called VDDIO or VDD):
  - the voltage range is between 1.7 V and 3.6 V
  - it supplies a part of the I/O ring, the embedded regulators and the system analog IPs as power management block and embedded oscillators
- VDD12o:
  - always-on digital power domain
  - this domain is generally supplied at 1.2 V during active phase of the device
  - this domain is supplied at 1.0V during low power mode (Deepstop)
- VDD12i:
  - interruptible digital power domain
  - this domain is generally supplied at 1.2 V during active phase of the device
  - this domain is shut down during low power mode (Deepstop).

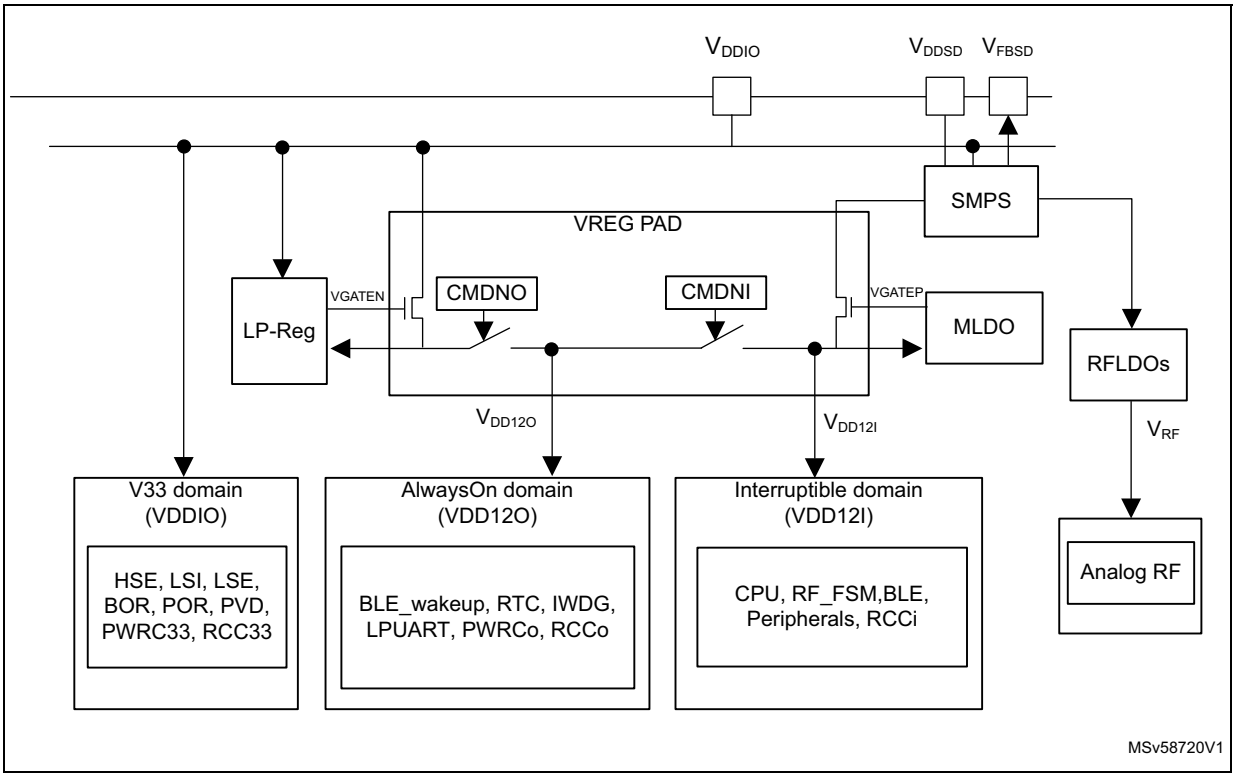
The digital power supplied are provided by different regulators:

- a main LDO (MLDO):
  - providing the 1.2 V from a 1.4-3.3 V input voltage
  - supplies both VDD12i and VDD12o when the device is active
  - is disabled during the low power mode (Deepstop)
- a low power LDO (LPREG):
  - stays enabled during both active and low power phases
  - providing a 1.0V voltage
  - is not connected to the digital domain when the device is active
  - is connected to the VDD12o domain during low power mode (Deepstop)
- a dedicated LDO (RFLDO) to provide a 1.2V to the analog RF block.

An embedded SMPS step down converter is available (inserted between the external power and the LDOs).

Figure 4 shows an overview of the different regulators and connections between the power supplies domains.

Figure 4. Power supply domains overview



5.3 Power voltage supervisor

The STM32WB09xE device embeds several power voltage monitoring:

- Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR)
- Programmable voltage detector (PVD)

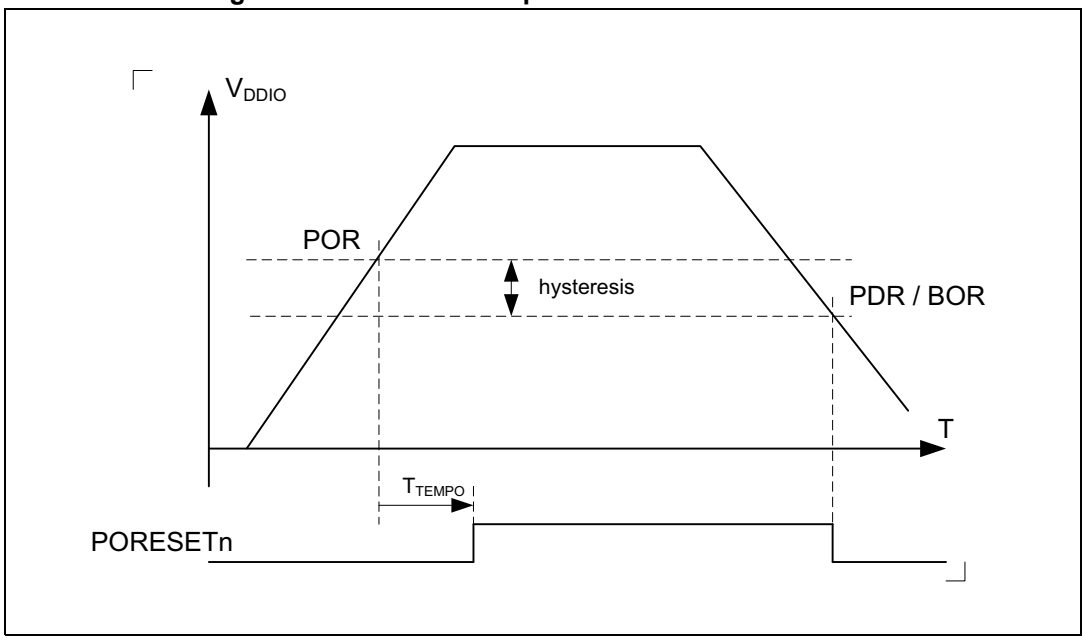
5.3.1 Power on reset POR / power down reset (PDR) / brown-out reset (BOR)

The device has an integrated power on reset / power down reset, coupled with a brown-out reset circuitry.

During power on, the device remains in reset mode as long as VDDIO is below a V_{POR} threshold (typically 1.60 V).

During power down, the PDR puts the device under reset when the supply voltage (VDD) drops below the V_{PDR} threshold (around 20mV below V_{POR}). The PDR feature is always enabled.

Figure 5. Power on reset/power down reset waveform



With typical values as follows:

- V_{POR}: 1.60 V
- hysteresis: 20 mV (so V_{PDR}: 1.58 V)
- T_{TEMPO}: 600 µs

The Brown-Out reset (BOR) generates a device reset when the power supply (VDD) drops under V_{PDR}.

This feature is always active except during the Shutdown mode where the software can decide to enable it or not (through PWRC_CR1.ENSDNBOR bit).

5.3.2 Programmable voltage detectior (PVD)

The PVD can be used to monitor:

- The VDDIO:
  - the VDDIO is compared to a programmed threshold (between 2.04 V and 2.94 V)
  - the threshold programming is done through PWRC_CR2.PVDLS[2:0] bit field
- An external analog input signal:
  - an external analog signal is compared to an internal VBGP (at 1.0V) voltage
  - the feature is selected through PWRC_CR2.PVDLS[2:0] bit field

The PVD can be enabled or disabled through PWRC_CR2.PVDE bit.

When the feature is enabled and the PVD measure a voltage below the comparator, a status flag is raised in the SYSCFG block that can generate an interrupt to the CPU if unmasked (see *Section 8.2.8: I/O Interrupt Status and Clear register (IO_ISCR)*).

5.4 Operating modes

The STM32WB09xE supports three main operating modes:

- Run mode
- Deepstop mode
- Shutdown mode

The transition from one mode to another is managed through a PMU state machine.

5.4.1 Run mode

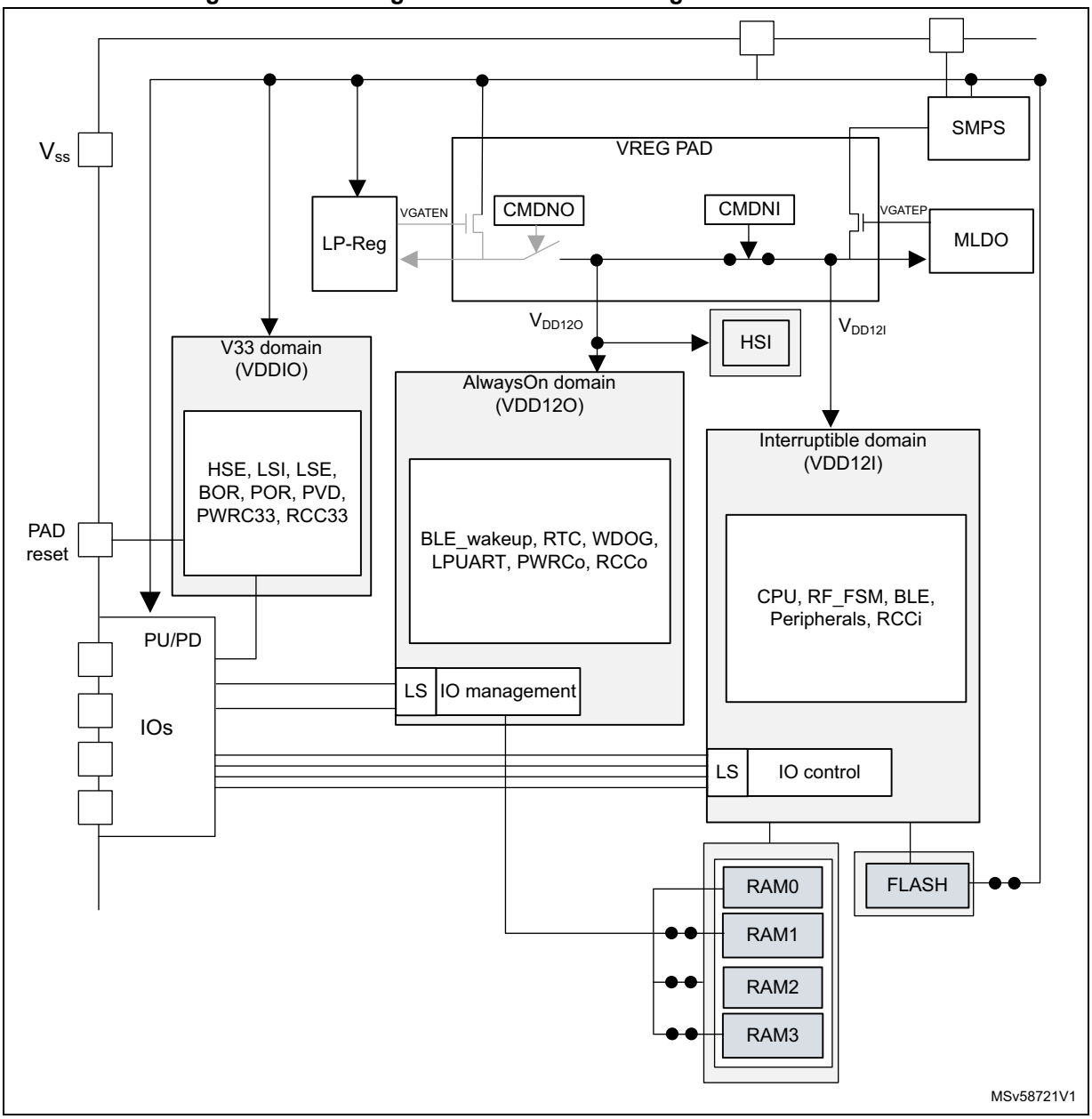
In Run mode:

- Both regulators (MLDO and LPREG) are enabled
- The MLDO provides the power supply for both VDD12i and VDD12o
- The system clock and bus clock are running
- The CPU and the radio can be used.

The power consumption may be reduced by gating the clock of the unused peripherals through the RCC clock enable registers (see *Section 6.6.18: RCC register map*).

Figure 6 shows the regulators and SMPS configuration in Run mode with a product with only 24 Kbytes of RAM.

Figure 6. Power regulators and SMPS configuration in Run mode



5.4.2 Deepstop mode

The Deepstop is the only low power mode of the STM32WB09xE allowing to restart from a saved context environment and go on running the application at wakeup.

The conditions to enter the Deepstop mode are:

- The radio (MR_BLE) is sleeping
- The CPU is sleeping (WFI with SLEEPDEEP information active)
- No unmasked wakeup sources is active (including those from a previous wakeup sequence for which the software did not clear the associated flag after wakeup)
- The system is clocked on the RC64MPLL (HSI or pll locked mode)
- The PWRC_CR1.LPMS bit is equal to 0.
- If PWRC_CR1.APC = 1, the I/O pull-ups/pull-downs are controlled by the PWRC_PUCRx/PWRC_PDCRx registers.
- Set GPIORET bit to enable GPIOs configuration retention for all I/Os (If DEEPSTOP2 bit is set, GPIORET must be reset, because SWJTAG must be available).

Note: If the MR_BLE is not used at all by the SoC (or not yet started), the following steps need to be done after any reset to allow low power modes (Deepstop and Shutdown):

  – Enable the MR_BLE clock by setting the RCC_APB2ENR.MRBLEEN bit.
  – Set the BLE_SLEEP_REQUEST_MODE.FORCE_SLEEPING bit inside the Wakeup block of the MR_BLE to have the MR_BLE IP requesting low power mode to the SoC.
  – Gate again the MR_BLE clock by clearing the RCC_APB2ENR.MRBLEEN bit.

In Deepstop mode:

- The system and bus clocks are stopped as the RC64MPLL block is OFF
- The VDD12i power domain is switched off
- The VDDI2o power domain is ON and supplied at 1.0 V
- the RAM0 bank is kept in retention
- The other RAM banks are in retention or not, depending on software choice in PWRC_CR2 register
- The slow clock can be running or stopped, depending on the software configuration present before Deepstop entry:
  – ON or OFF
  – LSE or LSI source
- The RTC, IWDG and LPUART stay active (if enabled and one slow clock source is ON)
- the MR_BLE wakeup block including its timer stay active (if enabled and one slow clock source is ON)
- All I/Os configurations can be retained and are able to be configured:
  – in output:
  a) Driving either a static low or high level if *Section 7.4: GPIO registers* properly programmed.
  b) Driving the slow clock information RCC_LCO on PA10 only if LCOEN bit is set.
  c) Driving the RTC_OUT on PA8 only if *Section 19.6.3: RTC control register (RTC_CR)* properly programmed and AF1 selected.
  – In input if *Section 7.4: GPIO registers* properly programmed.

A version of the Deepstop mode called Deepstop2 has been implemented to emulate the Deepstop mode without losing the debugger connection and breakpoints nor watchpoints.

- This variant can be selected by setting the PWRC_DBGR.DEEPSTOP2 bit.
- In this case, the Deepstop mode sequence (entry and exit) is done without shutting down the VDD12i power domain.

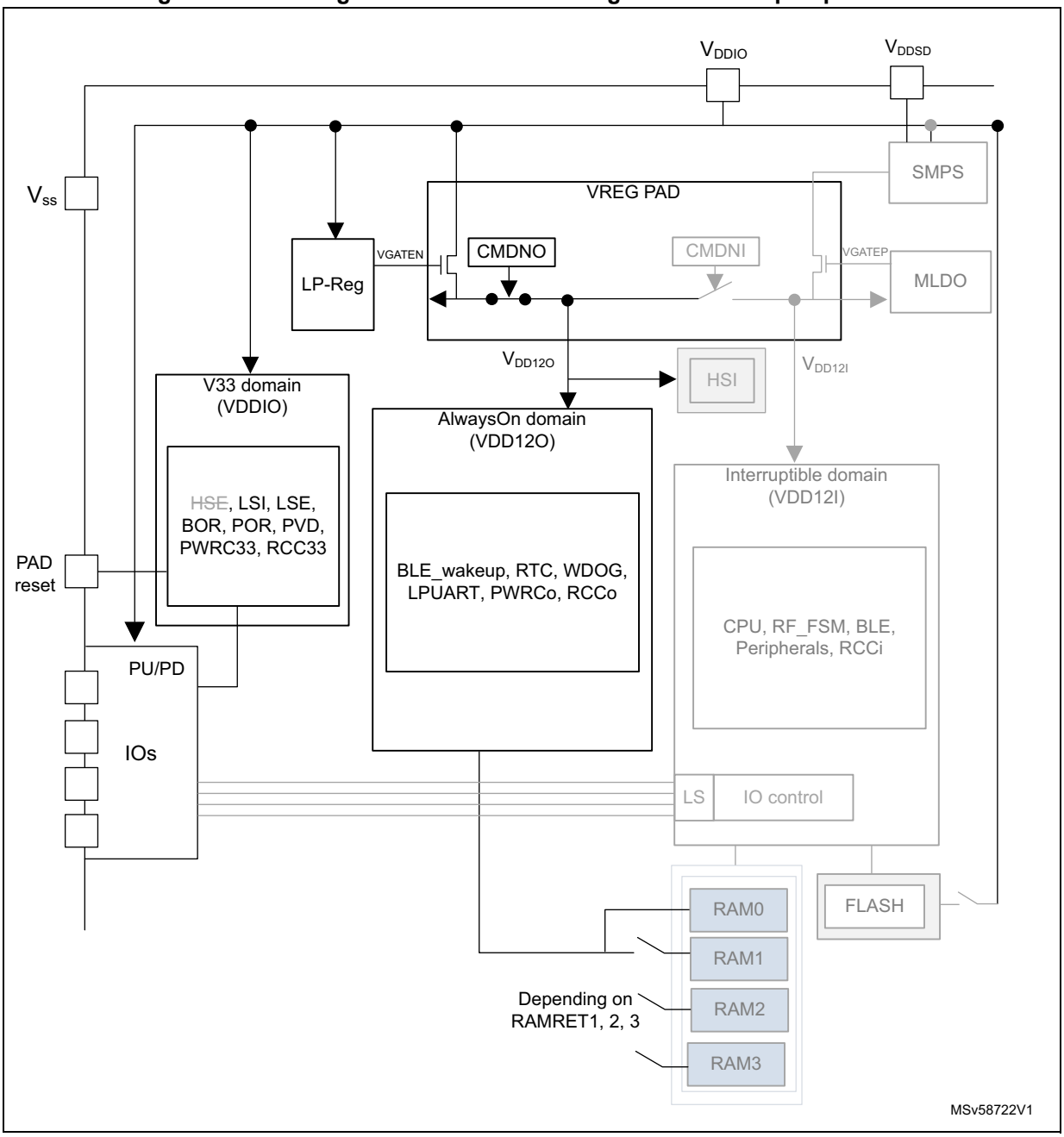
Possible wakeup sources:

- The MR_BLE block is able to generate two events to wakeup the system through its embedded wakeup timer running on slow clock:
  - Bluetooth LE IP wakeup time is reached
  - Host CPU wakeup time is reached
- The RTC is able to generate a wakeup event
- The LPUART is able to generate a wakeup event
- The IWDG is able to generate a reset event
- All I/Os are able to wakeup the system.

After wakeup from Deepstop, all the I/Os are in retention mode (except PA2 and PA3 in order to have SWJTAG available again); if DBGRET bit was set, before entering Deepstop mode, then at wakeup also PA2 and PA3 are in retention mode and SWJTAG is not available. To change I/Os configuration, when exiting Deepstop, it is necessary to reset GPIORET bit after having re-configured GPIO registers through *Section 7.4: GPIO registers* (this GPIO configuration, can be the same before entering Deepstop, or a new one). At wakeup, the hardware resources located in the VDD12i power domain are reset, the CPU reboots. The wakeup reason is visible in a PWRC register (see *Section 5.7.5: Status register 1 (PWRC_SR1)* for details).

Figure 7 shows the regulators and SMPS configuration in Deepstop mode, with a configuration requesting retention only on RAM1, RAM2 and RAM3 banks.

Figure 7. Power regulators and SMPS configuration in Deepstop mode(b)



b. Strikethrough text indicates that the feature is OFF in Deepstop mode.

5.4.3 Shutdown mode

The Shutdown mode is the least power consuming mode.

The conditions to enter the Shutdown mode are the same conditions needed to enter in Deepstop mode except that the PWRC_CR1.LPMS bit must be equal to 1 (PWRC_DBGR.DEEPSTOP2 bit must be maintained equal to 0).

In Shutdown mode:

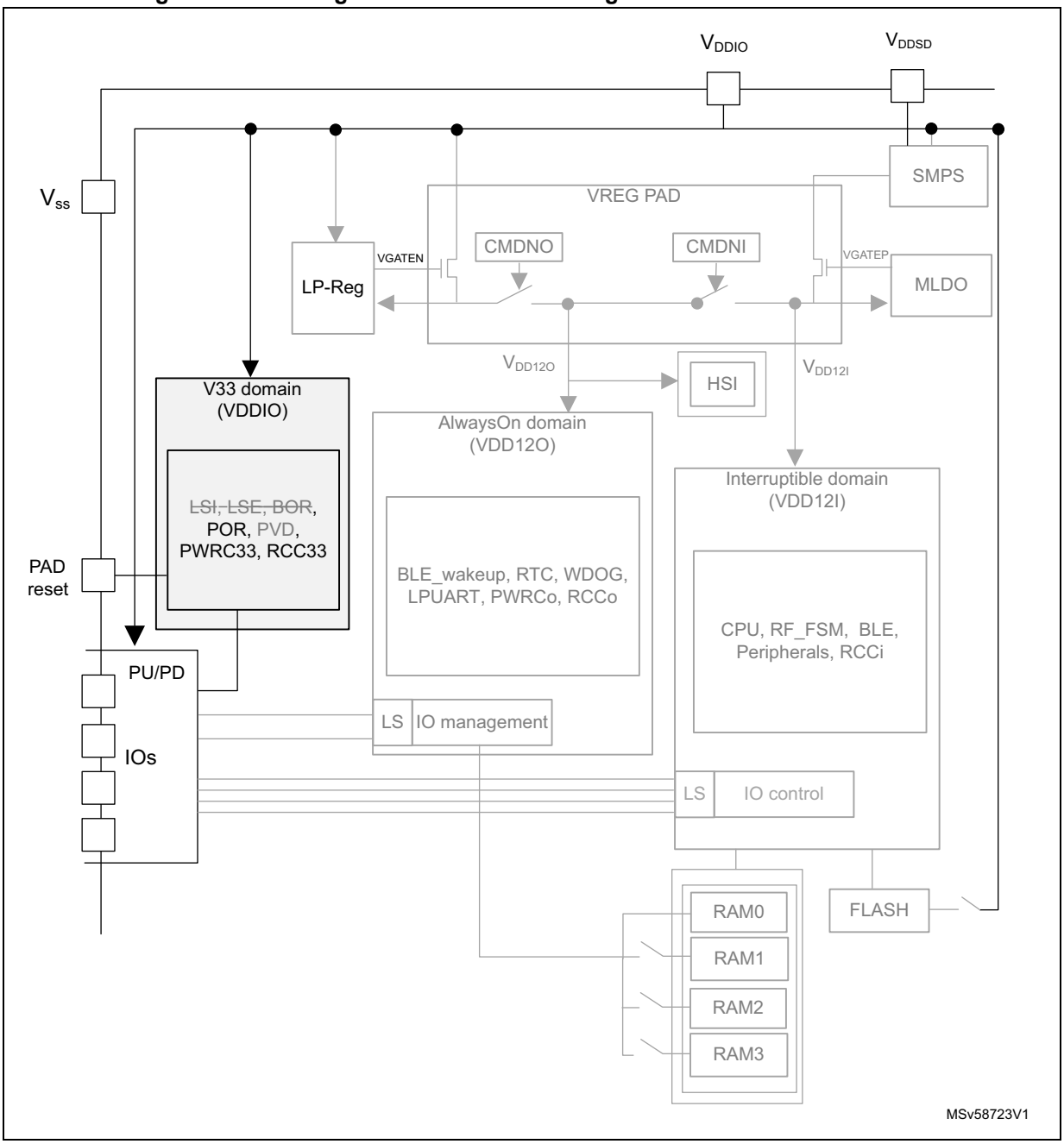
- The system is powered down as both regulators are OFF (so both VDD12i and VDD12o power domains are OFF).
- Only the VDDIO power domain is ON.
- All clocks are OFF (system and slow clock tree) as RC64MPLL, LSI and LSE are OFF.
- If PWRC_CR1.APC = 1, the I/O pull-ups/pull-downs are controlled by the PWRC_PUCRx/PWRC_PDCRx registers.
- The only wakeup sources are a low pulse on the RSTN pin or configurable pulse on PB0 pin.
- The only wakeup sources are a low pulse on the RSTN pin or configurable pulse or level on PB0 pin through *Shutdown I/O Wakeup Enable register (PWRC_SDWN_WUEN)* and *Shutdown I/O Wakeup Polarity register (PWRC_SDWN_WUPOL)*.

A Shutdown exit is similar to a POR startup of the board. The associated reset reason is the PORRSTF flag or EWUF flag (see *Section 6.6.15: Clock control and reset status register (RCC_CSR)* for reset reason flags detail or *Shutdown I/O Wakeup Flag register (PWRC_SDWN_WUF)* for PB0 wakeup flag).

The BOR feature may be enabled or disabled during Shutdown through the PWRC_CR1.ENSDNBOR bit).

Figure 8 shows the regulators and SMPS configuration in Shutdown mode, configured with the BOR reset disabled.

Figure 8. Power regulators and SMPS configuration in Shutdown mode(c)

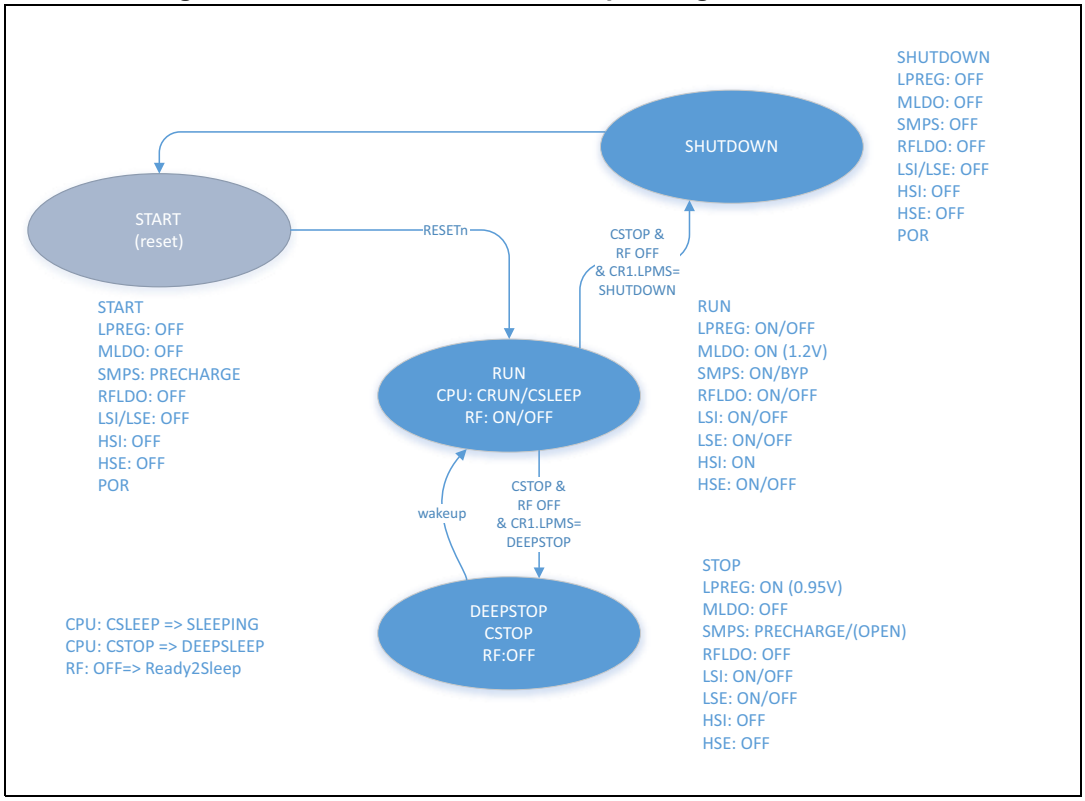


c. Strikethrough text indicates that the feature is OFF in Shutdown mode.

5.4.4 Operating modes transition management

The PWRC block manages the switches from an operating mode to another through a state machine.

Figure 9. PWRC state machine for operating modes transition



5.5 SMPS step down regulator

The STM32WB09xE SMPS is a 20 mA output step down SMPS (switch mode power supply) converter.

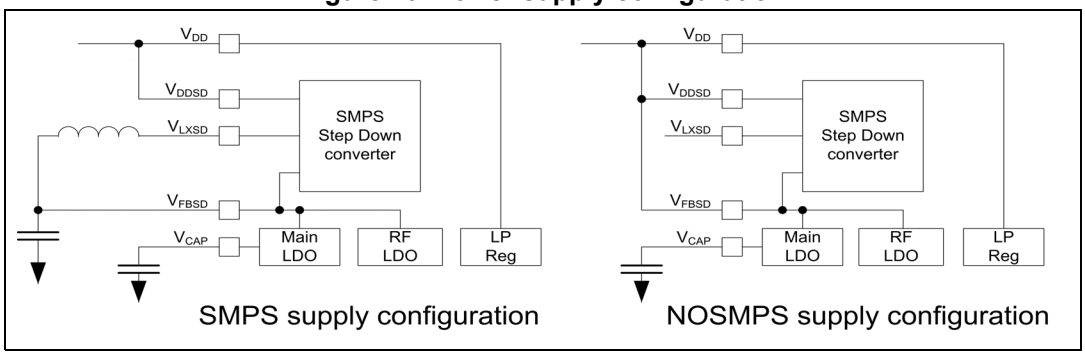
The SMPS output voltage can be programmed from 1.2 V to 1.9 V. It is internally clock at 4 MHz or 8 MHz.

The SMPS can be in different configurations:

- ON:
  - the V_{FBSD} pin of the SMPS outputs a regulated voltage (from 1.2 V to 1.9 V)
  - the SMPS needs a clock
- OFF:
  - the V_{FBSD} pin has to be forced externally with VDDIO
  - the SMPS does not need a clock
- PRECHARGE (also called BYPASS):
  - the V_{FBSD} pin output the VDDIO without regulation
  - the SMPS does not need a clock
  - the SMPS current can be limited programming *Control register 5 (PWRC_CR5)*.
- OPEN:
  - the V_{FBSD} pin is floating
  - the SMPS does not need a clock.

Except for the configuration SMPS OFF, a L/C BOM must be present on the board and connected to the V_{FBSD} pad (see *Figure 10*).

Figure 10. Power supply configuration



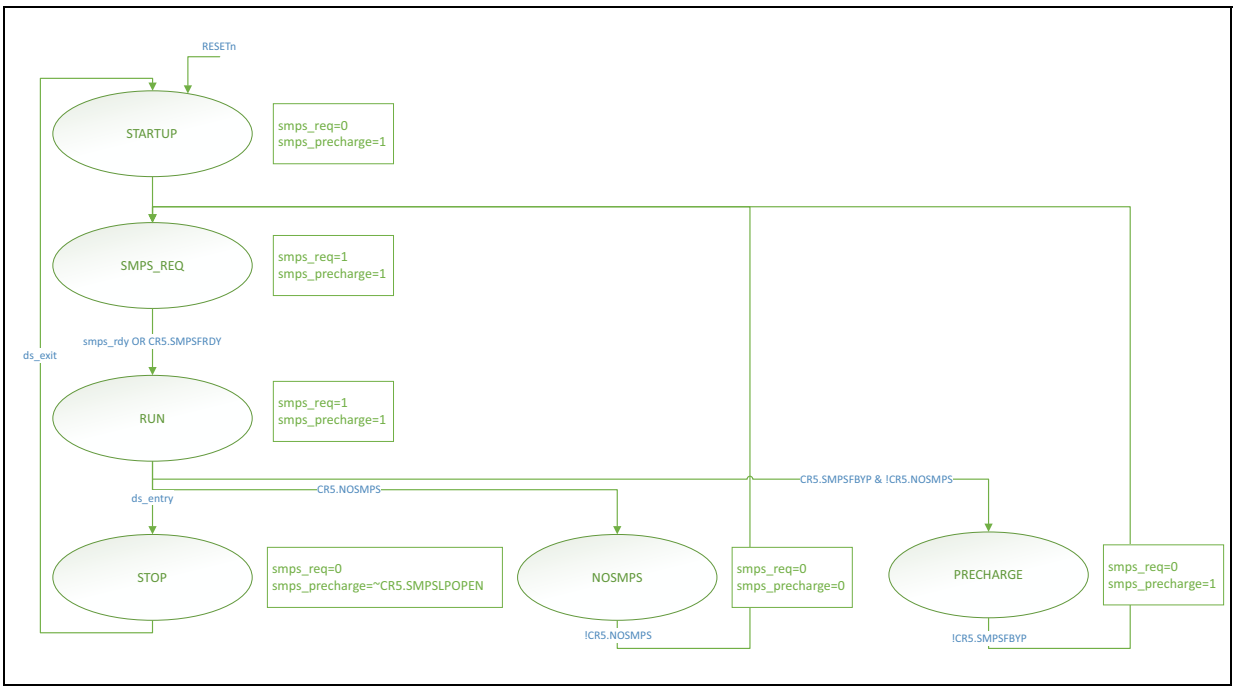
The user must configure the provides the PWRC_CR5.SMPSBOMSEL[1:0] according to the BOM implemented on his board. The value to program is indicated in *Table 11*.

Table 11. SMPS BOM information

BOM	Inductance (L)	Output capacitance (C)	SMPSBOMSEL[1:0]
BOM1	1.5 µH	2.2 µF	00
BOM2	2.2 µH	4.7 µF	01
BOM3	10 µH	4.7 µF	10

The SMPS is managed by the PWRC through a state machine shown in *Figure 11*.

Figure 11. PWRC SMPS state machine overview



After a Power On reset sequence, the SMPS FSM always goes up to Run state.

From there, the SMPS FSM can stay in three states (others are transition states):

- Run:
  - the SMPS is ON in a run mode
  - the SMPS clock is running
  - the V_{FBSD} is regulated and voltage amplitude is the one programmed in the PWRC_CR5.SMPSLVL bit field
  - a L/C BOM is present on the board and is connected on the VFBSD pad of the STM32WB09xE (see *Figure 10*)
- NOSMPS (if the software configures PWRC_CR5.NOSMPS=1):
  - the SMPS is OFF
  - the SMPS clock is stopped
  - the V_{FBSD} is directly connected to the VDDIO through the VFBSD pad of the STM32WB09xE (see *Figure 10*)
  - the PWRC does not control any specific sequencing on the SMPS during low power entry/exit phases
- PRECHARGE, also called BYPASS, (if the software configures PWRC_CR5.SMPSFBYP=1):
  - the SMPS is ON in a precharge mode
  - the SMPS clock is stopped
  - the V_{FBSD} is the VDDIO voltage crossing the SMPS block
  - the PWRC does not control any specific sequencing on the SMPS during low power entry/exit phases
  - this mode is compliant with a L/C BOM connected on the VFBSD pad of the STM32WB09xE.

When the device enters in Deepstop mode, the PWRC automatically switches the SMPS from Run to Deepstop mode which can be:

- if PWRC_CR5.SMPSLPOPEN = 0: SMPS output is the VDDIO (as in PRECHARGE FSM state)
- if PWRC_CR5.SMPSLPOPEN = 1: the SMPS output is floating.

5.6 I/O pull-ups/pull-downs during low power mode

When PWRC_CR1.APC is set (default configuration), the pull-up/-down of the IOs is controlled by the PWRC_PUCRx and PWRC_PDCRx registers of the PWRC block, instead of GPIOx_PUPDR register of the GPIO block. This feature avoids glitches on the lines by keeping the same pull-up/-down configuration during all the supported operating modes.

5.7 PWRC registers

Note: all the PWRC APB registers are only 16-bit registers. The 16 MSB bits are always stuck to 0.

5.7.1 Control register 1 (PWRC_CR1)

This register control the BOR in Shutdown, the low power mode selection and the IO control owner.

Address offset: 0x00

Reset value: 0x0000 0114

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APC	IBIAS_ RUN_S TATE	IBIAS_ RUN_A UTO	ENSDN BOR	LPMS
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **APC**: Apply pull-up/-down configuration from PWRC or GPIO register.
0: The GPIOx_PUPDR of the GPIO block are used to control the product pull-up/-down of the IOs.
1: The PWRC_PUCRx and PWRC_PDCRx of the PWRC block are used to control the product pull-up/-down of the IOs (default).

Bit 3 **IBIAS_RUN_STATE**: Enable/Disable IBIAS during Run mode when automatic mode is disabled.
– 0: IBIAS control is disabled (default).
– 1: IBIAS control is enabled.

Bit 2 **IBIAS_RUN_AUTO**: IBIAS_RUN_AUTO: Enable automatic IBIAS control during Run or Deepstop mode.
– 0: IBIAS control is manual (and controlled by IBIAS_RUN_STATE register)
– 1: IBIAS control is automatic (default).

Bit 1 **ENSDNBOR**: Enable BOR reset supervising during Shutdown mode.
0: No BOR is monitored during Shutdown mode (default).
1: BOR is monitored during Shutdown mode (a POR reset happens if VDDIO goes below 1.58V during Shutdown mode).
Note: Enabling this feature prevents blocking the device if VDDIO goes below supported voltages during Shutdown. However, it adds an overconsumption.

Bit 0 **LPMS**: Low Power Mode Selection.
This bit defines if the device enters Deepstop or Shutdown mode when both CPU and MR_BLE requests a low power mode entry.
0: Deepstop mode (default).
1: Shutdown mode.

Note: It is s mandatory to ensure that PWRC_DBGR.DEEPSTOP2 *bit is reset when LPMS bit is set before entering Shutdown mode.*

5.7.2 Control register 2 (PWRC_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIORET	RAMRET3	RAMRET2	RAMRET1	DBGRET	PVDLS[2:0]			PVDE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 (1) **GPIORET**: GPIO retention enable.
0: GPIOs don't retain their configuration during and exiting Deepstop (default)
1: GPIOs retain their configuration during and exiting Deepstop.

Bit 7 **RAMRET3**: enable the RAM3 bank retention in Deepstop mode.
0: RAM3 bank is not retained during Deepstop mode (default).
1: RAM3 bank is retained during Deepstop mode.

Bit 6 **RAMRET2**: enable the RAM2 bank retention in Deepstop mode.
0: RAM2 bank is not retained during Deepstop mode (default).
1: RAM2 bank is retained during Deepstop mode.

Bit 5 **RAMRET1**: enable the RAM1 bank retention in Deepstop mode.
0: RAM1 bank is not retained during Deepstop mode (default).
1: RAM1 bank is retained during Deepstop mode.

Bit 4 **DBGRET**: PA2 and PA3 retention enable after Deepstop
0: PA2, PA3 GPIOs don't retain their configuration exiting from Deepstop (default).
1: PA2, PA3 GPIOs retain their configuration exiting from Deepstop.

Bits 3:1 **PVDLS[2:0]**: Programmable Voltage Detector level selection:
000: 2.04V - Lowest level
001: 2.22V
010: 2.36V
011: 2.53V
100: 2.66V
101: 2.81V
110: 2.94V - Highest level
111: Reserved

Bit 0 **PVDE**: Programmable Voltage Detector enable.
0: the power voltage detector feature is disabled (default).
1: the power voltage detector feature is enabled.

Note: it's mandatory to ensure GPIORET bit is set before entering Deepstop unless the DEEPSTOP2 bit is set.

5.7.3 Control register 3 (PWRC_CR3)

This register manages the selection of the wakeup sources to get out of Deepstop mode.

Note: All wakeup sources are disabled by default after reset.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIWL	EIWL2	EWBLE HCPU	EWBLE	EWU11	EWU10	EWU9	EWU8	EWU7	EWU6	EWU5	EWU4	EWU3	EWU2	EWU1	EWU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EIWL**: Enable wakeup on Internal event (RTC).
0: Wakeup on internal line is disabled (default).
1: Wakeup on internal line is enabled.

Bit 14 **EIWL2**: Enable wakeup on Internal event (LPUART).
0: Wakeup on internal line is disabled (default).
1: Wakeup on internal line is enabled.

Bit 13 **EWBLEHCPU**: Enable wakeup on Bluetooth LE Host CPU event.
0: Wakeup on Bluetooth LE Host CPU line is disabled (default).
1: Wakeup on Bluetooth LE Host CPU line is enabled.

Bit 12 **EWBLE**: Enable wakeup on Bluetooth LE event.
0: Wakeup on Bluetooth LE line is disabled (default).
1: Wakeup on Bluetooth LE line is enabled.

Bit 11 **EWU11**: Enable wakeup on PA11 I/O event.
0: Wakeup on PA11 I/O line is disabled (default).
1: Wakeup on PA11 I/O line is enabled.

Bit 10 **EWU10**: Enable wakeup on PA10 I/O event.
0: Wakeup on PA10 I/O line is disabled (default).
1: Wakeup on PA10 I/O line is enabled.

Bit 9 **EWU9**: Enable wakeup on PA9 I/O event.
0: Wakeup on PA9 I/O line is disabled (default).
1: Wakeup on PA9 I/O line is enabled.

Bit 8 **EWU8**: Enable wakeup on PA8 I/O event.
0: Wakeup on PA8 I/O line is disabled (default).
1: Wakeup on PA8 I/O line is enabled.

Bit 7 **EWU7**: Enable wakeup on PB7 I/O event.
0: Wakeup on PB7 I/O line is disabled (default).
1: Wakeup on PB7 I/O line is enabled.

Bit 6 **EWU6**: Enable wakeup on PB6 I/O event.
0: Wakeup on PB6 I/O line is disabled (default).
1: Wakeup on PB6 I/O line is enabled.

Bit 5 **EWU5**: Enable wakeup on PB5 I/O event.
0: Wakeup on PB5 I/O line is disabled (default).
1: Wakeup on PB5 I/O line is enabled.

Bit 4 **EWU4**: Enable wakeup on PB4 I/O event.
0: Wakeup on PB4 I/O line is disabled (default).
1: Wakeup on PB4 I/O line is enabled.

Bit 3 **EWU3**: Enable wakeup on PB3 I/O event.
0: Wakeup on PB3 I/O line is disabled (default).
1: Wakeup on PB3 I/O line is enabled.

Bit 2 **EWU2**: Enable wakeup on PB2 I/O event.
0: Wakeup on PB2 I/O line is disabled (default).
1: Wakeup on PB2 I/O line is enabled.

Bit 1 **EWU1**: Enable wakeup on PB1 I/O event.
0: Wakeup on PB1 I/O line is disabled (default).
1: Wakeup on PB1 I/O line is enabled.

Bit 0 **EWU0**: Enable wakeup on PB0 I/O event.
0: Wakeup on PB0 I/O line is disabled (default).
1: Wakeup on PB0 I/O line is enabled.

5.7.4 Control register 4 (PWRC_CR4)

This register manages the polarity for the I/Os wakeup sources to get out of Deepstop mode.

Note: The wakeup event are only edge detection, not level detection.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WUP11	WUP10	WUP9	WUP8	WUP7	WUP6	WUP5	WUP4	WUP3	WUP2	WUP1	WUP0
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **WUP11**: Wakeup polarity for PA11 I/O.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 10 **WUP10**: Wakeup polarity for PA10 I/O.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 9 **WUP9**: Wakeup polarity for PA9 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 8 **WUP8**: Wakeup polarity for PA8 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 7 **WUP5**: Wakeup polarity for PB5 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 6 **WUP6**: Wakeup polarity for PB6 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 5 **WUP5**: Wakeup polarity for PB5 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 4 **WUP4**: Wakeup polarity for PB4 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 3 **WUP3**: Wakeup polarity for PB3 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 2 **WUP2**: Wakeup polarity for PB2 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 1 **WUP1**: Wakeup polarity for PB1 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 0 **WUP0**: Wakeup polarity for PB0 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

5.7.5 Status register 1 (PWRC_SR1)

This register provides the information concerning which source woke up the device after a Deepstop.

*A*ddress offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWUF	IWUF2	WBLE HCPUF	WBLEF	WUF11	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
r	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **IWUF**: Internal wakeup flag (RTC).
0: no wakeup from RTC occurred since last clear.
1: a wakeup from RTC occurred since last clear.
Note: The user must clear the RTC wakeup flag inside the RTC IP to clear this bit (mirror of the RTC wakeup line on the PWRC block).

Bit 14 **IWUF2**: Internal wakeup 2 flag (LPUART).
0: no wakeup from LPUART occurred since last clear.
1: a wakeup from LPUART occurred since last clear.
Cleared by writing 1 in this bit.
Note: The user must clear before LPUART wakeup flag inside the LPUART IP to clear this bit.

Bit 13 **WBLEHCPUF**: Bluetooth LE Host CPU wakeup flag.
0: no wakeup from Bluetooth LE Host CPU occurred since last clear.
1: a wakeup from Bluetooth LE Host CPU occurred since last clear.
Cleared by writing 1 in this bit.

Bit 12 **WBLEF**: Bluetooth LE wakeup flag.
0: no wakeup from Bluetooth LE occurred since last clear.
1: a wakeup from Bluetooth LE occurred since last clear.
Cleared by writing 1 in this bit.

Bit 11 **WUF11**: PA11 I/O wakeup flag.
0: no wakeup from PA11 I/O occurred since last clear.
1: a wakeup from PA11 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 10 **WUF10**: PA10 I/O wakeup flag.
0: no wakeup from PA10 I/O occurred since last clear.
1: a wakeup from PA10 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 9 **WUF9**: PA9 I/O wakeup flag.
0: no wakeup from PA9 I/O occurred since last clear.
1: a wakeup from PA9 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 8 **WUF8**: PA8 I/O wakeup flag.
0: no wakeup from PA8 I/O occurred since last clear.
1: a wakeup from PA8 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 7 **WUF7**: PB7 I/O wakeup flag.
0: no wakeup from PB7 I/O occurred since last clear.
1: a wakeup from PB7 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 6 **WUF6**: PB6 I/O wakeup flag.
0: no wakeup from PB6 I/O occurred since last clear.
1: a wakeup from PB6 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 5 **WUF5**: PB5 I/O wakeup flag.
0: no wakeup from PB5 I/O occurred since last clear.
1: a wakeup from PB5 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 4 **WUF4**: PB4 I/O wakeup flag.
0: no wakeup from PB4 I/O occurred since last clear.
1: a wakeup from PB4 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 3 **WUF3**: PB3 I/O wakeup flag.
0: no wakeup from PB3 I/O occurred since last clear.
1: a wakeup from PB3 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 2 **WUF2**: PB2 I/O wakeup flag.
0: no wakeup from PB2 I/O occurred since last clear.
1: a wakeup from PB2 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 1 **WUF1**: PB1 I/O wakeup flag.
0: no wakeup from PB1 I/O occurred since last clear.
1: a wakeup from PB1 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 0 **WUF0**: PB0 I/O wakeup flag.
0: no wakeup from PB0 I/O occurred since last clear.
1: a wakeup from PB0 I/O occurred since last clear.
Cleared by writing 1 in this bit.

5.7.6 Status register 2 (PWRC_SR2)

This register provides some status flags related to the Power Voltage Detector and the SMPS blocks.

*A*ddress offset: 0x14

Reset value: 0x0000 -3-6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOBOOTVAL[3:0]				PVDO	Res.	Res.	REGLPS	IOBOOTVAL2[3:0]				Res.	SMPSRDY	SMPSENR	SMPSBYPR
r	r	r	r	r			r	r	r	r	r		r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IOBOOTVAL**: I/Os value latched at POR.
bit 3: PA11 input value latched at POR,
bit 2: PA10 input value latched at POR,
bit 1: PA9 input value latched at POR,
bit 0: PA8 input value latched at POR.
Note: This information may be used but he boot loader to manage boot on serial interfaces for instance.

Bit 11 **PVDO**: Power voltage Detector Output.
When the Power voltage Detector is enabled (PWRC_CR2.PVDE=1), this bit indicates when the VDDIO is lower than the selected threshold (through PWRC_CR2.PVDLS bit field).
0: The VDDIO is not lower than threshold or PVD feature is not enabled.
1: The VDDIO is lower than the selected threshold.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **REGLPS**: Low Power regulator ready status.
0: The Low Power regulator is not ready.
1: The Low Power regulator is ready.

Bits 7:4 **IOBOOTVAL2**: I/Os value latched at POR.
bit 3: PB15 input value latched at POR,
bit 2: PB14 input value latched at POR,
bit 1: PB13 input value latched at POR,
bit 0: PB12 input value latched at POR.
Note: This information may be used but he boot loader to manage boot on serial interfaces for instance.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SMPSRDY**: SMPS ready status.
0: SMPS regulator is not ready.
1: SMPS regulator is ready.

Bit 1 **SMPSENR**: SMPS Run mode status.
This bit mirrors the internal ENABLE_3V3 control signal connected to the SMPS and driven by the hardware.
0: SMPS regulator is not regulating (in PRECHARGE or NOSMPS mode).
1: SMPS regulator is in Run mode.

Bit 0 **SMPSBYPR**: SMPS PRECHARGE mode status.
This bit mirrors the PRECHARGE control state of the SMPS.
0: SMPS regulator is not in PRECHARGE mode.
1: SMPS regulator is in PRECHARGE mode (VSMPS connected to VDDIO)

5.7.7 Control register 5 (PWRC_CR5)

This register is used to configure the SMPS.

*A*ddress offset: 0x1C

Reset value: 0x0000 6014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SMPS_PRECH_CUR_SEL[1:0]		CLKDETR_DISABLE	SMPS_ENA_DCM	NOSMPS	SMPSFBYP	SMPSLPOPEN	Res.	Res.	SMPSBOMSEL[1:0]		SMPSLVL[3:0]			
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:13 **SMPS_PRECH_CUR_SEL[1:0]**: Select SMPS PRECHARGE limit current.

00: 2.5 mA
01: 5 mA
10: 10 mA
11: 20 mA (default)

Bit 12 **CLKDETR_DISABLE**: disable the SMPS clock detection

The SMPS clock detection enables an automatic SMPS bypass switching in case of unexpected loss of the SMPS clock.

0: SMPS clock detection mechanism enabled (default).
1: SMPS clock detection mechanism disabled.

Bit 11 **SMPS_ENA_DCM**: Discontinuous conduction mode enable.

0: SMPS DCM is disabled (default).
1: SMPS DCM is enabled.

Bit 10 **NOSMPS**: No SMPS mode.

0: SMPS is enabled (default).
1: SMPS is disabled.

Note: this configuration (SMPS disabled) should be used only when the SMPS_FB pad is directly connected to the VBATT (external voltage), without L/C BOM.

Bit 9 **SMPSFBYP**: Force the SMPS in PRECHARGE mode.

0: no effect (default).
1: SMPS is disabled and bypassed.

Note: when this bit is set, the VSMPS output is connected to the VDDIO. The actual state of the SMPS is visible in the SMPS mode status bits in PWRC_SR2 register.

Bit 8 **SMPSLPOPEN**: Select OPEN mode instead of PRECHARGE mode for the SMPS during Deepstop.

0: in Deepstop, the SMPS is in PRECHARGE mode with output connected to VDDIO (default).
1: in Deepstop, the SMPS is disabled with floating output.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SMPSBOMSEL[1:0]**: Select the SMPS BOM.

00: BOM1
01: BOM2 (default)
10: BOM3
11: Not applicable

Note: BOM correspondence/details is available in Table 11: SMPS BOM information.

Bits 3:0 **SMPSLVL[3:0]**: Select the SMPS output voltage level.

This bit field selects the SMPS voltage output level with a granularity of ~50mV.

The SMPS output voltage level, V_{SMPS}, must be configured such that VBAT- V_{SMPS} ≥ 0.2V

0000: 1.2 V
0001: 1.25 V
0010: 1.3 V
0011: 1.35 V
0100: 1.4 V
0101: 1.45 V
0110: 1.5 V
0111: 1.55 V
1000: 1.6 V
1001: 1.65 V
1010: 1.7 V
1011: 1.75 V
1100: 1.8 V
1101: 1.85 V
1110: 1.9 V
1111: 1.9 V

Warning: The SMPS output voltage must not be changed of more than one step while the SMPS is in use. The sequence to reprogram a new SMPS output voltage is described in *Section 5.8.2: SMPS output level re-programming*.

5.7.8 I/O Port A pull-up control register (PWRC_PUCRA)

This register is used to control the pull-up for the PA0 to PA15 I/O when the PWRC_CR1.APC bit is set.

Caution: If both pull-up and pull-down are enabled in the PWRC_PUCRA and PWRC_PDCRA registers for an I/O, then pull-down is applied.

The user must take care to disable the pull on I/O programmed in Analog mode.

Address offset: 0x20

Reset value: 0x0000 0F07

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PUA11	PUA10	PUA9	PUA8	Res.	Res.	Res.	Res.	PUA3	PUA2	PUA1	PUA0
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **PUA11**: Pull-up enable for PA11 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 10 **PUA10**: Pull-up enable for PA10 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 9 **PUA9**: Pull-up enable for PA9 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 8 **PUA8**: Pull-up enable for PA8 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **PUA3**: Pull-up enable for PA3 I/O.
0: No pull-up (default).
1: Pull-up enabled.

Bit 2 **PUA2**: Pull-up enable for PA2 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 1 **PUA1**: Pull-up enable for PA1 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 0 **PUA0**: Pull-up enable for PA0 I/O.
0: No pull-up.
1: Pull-up enabled (default).

5.7.9 I/O Port A pull-down control register (PWRC_PDCRA)

This register is used to control the pull-down for the PA0 to PA15 I/O when the PWRC_CR1.APC bit is set.

Caution: If both pull-up and pull-down are enabled in the PWRC_PUCRA and PWRC_PDCRA registers for an I/O, then pull-down is applied.

The user must take care to disable the pull on I/O programmed in Analog mode.

Address offset: 0x24

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PDA11	PDA10	PDA9	PDA8	Res.	Res.	Res.	Res.	PDA3	PDA2	PDA1	PDA0
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **PDA11**: pull-down enable for PA11 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 10 **PDA10**: pull-down enable for PA10 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 9 **PDA9**: pull-down enable for PA9 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 8 **PDA8**: pull-down enable for PA8 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **PDA3**: pull-down enable for PA3 I/O.
0: No pull-down.
1: pull-down enabled (default).

Bit 2 **PDA2**: pull-down enable for PA2 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 1 **PDA1**: pull-down enable for PA1 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 0 **PDA0**: pull-down enable for PA0 I/O.
0: No pull-down (default).
1: pull-down enabled.

5.7.10 I/O Port B pull-up control register (PWRC_PUCRB)

This register is used to control the pull-up for the PB0 to PB15 I/O when the PWRC_CR1.APC bit is set.

Caution: If both pull-up and pull-down are enabled in the PWRC_PUCRA and PWRC_PDCRA registers for an I/O, then pull-down is applied.

The user must take care to disable the pull on I/O programmed in Analog mode.

*A*ddress offset: 0x28

Reset value: 0x0000 F0FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUB15	PUB14	PUB13	PUB12	Res.	Res.	Res.	Res.	PUB7	PUB6	PUB5	PUB4	PUB3	PUB2	PUB1	PUB0
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PUB15**: Pull-up enable for PB15 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 14 **PUB14**: Pull-up enable for PB14 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 13 **PUB13**: Pull-up enable for PB13 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 12 **PUB12**: Pull-up enable for PB12 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **PUB7**: Pull-up enable for PB7 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 6 **PUB6**: Pull-up enable for PB6 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 5 **PUB5**: Pull-up enable for PB5 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 4 **PUB4**: Pull-up enable for PB4 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 3 **PUB3**: Pull-up enable for PB3 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 2 **PUB2**: Pull-up enable for PB2 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 1 **PUB1**: Pull-up enable for PB1 I/O.
0: No pull-up.
1: Pull-up enabled (default).

Bit 0 **PUB0**: Pull-up enable for PB0 I/O.
0: No pull-up.
1: Pull-up enabled (default).

5.7.11 I/O Port B pull-down control register (PWRC_PDCRB)

This register is used to control the pull-down for the PB0 to PB15 I/O when the PWRC_CR1.APC bit is set.

Caution: If both pull-up and pull-down are enabled in the PWRC_PUCRA and PWRC_PDCRA registers for an I/O, then pull-down is applied.

The user must take care to disable the pull on I/O programmed in Analog mode.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDB15	PDB14	PDB13	PDB12	Res.	Res.	Res.	Res.	PDB7	PDB6	PDB5	PDB4	PDB3	PDB2	PDB1	PDB0
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PDB15**: pull-down enable for PB15 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 14 **PDB14**: pull-down enable for PB14 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 13 **PDB13**: pull-down enable for PB13 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 12 **PDB12**: pull-down enable for PB12 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **PDB7**: pull-down enable for PB7 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 6 **PDB6**: pull-down enable for PB6 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 5 **PDB5**: pull-down enable for PB5 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 4 **PDB4**: pull-down enable for PB4 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 3 **PDB3**: pull-down enable for PB3 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 2 **PDB2**: pull-down enable for PB2 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 1 **PDB1**: pull-down enable for PB1 I/O.
0: No pull-down (default).
1: pull-down enabled.

Bit 0 **PDB0**: pull-down enable for PB0 I/O.
0: No pull-down (default).
1: pull-down enabled.

5.7.12 Control register 6 (PWRC_CR6)

This register manages the selection of the wakeup sources to get out of Deepstop mode.

Note: All wakeup sources are disabled by default after reset.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU19	EWU18	EWU17	EWU16	EWU15	EWU14	EWU13	EWU12
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **EWU19**: Enable wakeup on PB15 I/O event.
0: Wakeup on PB15 I/O line is disabled (default).
1: Wakeup on PB157 I/O line is enabled.

Bit 6 **EWU18**: Enable wakeup on PB14 I/O event.
0: Wakeup on PB14 I/O line is disabled (default).
1: Wakeup on PB14 I/O line is enabled.

Bit 5 **EWU17**: Enable wakeup on PB13 I/O event.
0: Wakeup on PB13 I/O line is disabled (default).
1: Wakeup on PB13 I/O line is enabled.

Bit 4 **EWU16**: Enable wakeup on PB12 I/O event.
0: Wakeup on PB12 I/O line is disabled (default).
1: Wakeup on PB12 I/O line is enabled.

Bit 3 **EWU15**: Enable wakeup on PA3 I/O event.
0: Wakeup on PA3 I/O line is disabled (default).
1: Wakeup on PA3 I/O line is enabled.

Bit 2 **EWU14**: Enable wakeup on PA2 I/O event.
0: Wakeup on PA2 I/O line is disabled (default).
1: Wakeup on PA2 I/O line is enabled.

Bit 1 **EWU13**: Enable wakeup on PA1 I/O event.
0: Wakeup on PA1 I/O line is disabled (default).
1: Wakeup on PA1 I/O line is enabled.

Bit 0 **EWU12**: Enable wakeup on PA0 I/O event.
0: Wakeup on PA0 I/O line is disabled (default).
1: Wakeup on PA0 I/O line is enabled.

5.7.13 Control register 7 (PWRC_CR7)

This register manages the polarity for the I/Os wakeup sources to get out of Deepstop mode.

Note: The wakeup event are only edge detection, not level detection.

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUP19	WUP18	WUP17	WUP16	WUP15	WUP14	WUP13	WUP12
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 7 **WUP19**: Wakeup polarity for PB15 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 6 **WUP18**: Wakeup polarity for PB14 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 5 **WUP17**: Wakeup polarity for PB13 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 4 **WUP16**: Wakeup polarity for PB12 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 3 **WUP15**: Wakeup polarity for PA3 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 2 **WUP14**: Wakeup polarity for PA2 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 1 **WUP13**: Wakeup polarity for PA1 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

Bit 0 **WUP12**: Wakeup polarity for PA0 IO event.
0: Detection of wakeup event on rising edge (default).
1: Detection of wakeup event on falling edge.

5.7.14 Status register 3 (PWRC_SR3)

This register provides the information concerning which source woke up the device after a Deepstop.

*A*ddress offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF19	WUF18	WUF17	WUF16	WUF15	WUF14	WUF13	WUF12
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:6 Reserved, must be kept at reset value.

Bit 7 **WUF19**: PB15 I/O wakeup flag.
0: no wakeup from PB15 I/O occurred since last clear.
1: a wakeup from PB15 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 6 **WUF18**: PB14 I/O wakeup flag.
0: no wakeup from PB14 I/O occurred since last clear.
1: a wakeup from PB14 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 5 **WUF17**: PB13 I/O wakeup flag.
0: no wakeup from PB13 I/O occurred since last clear.
1: a wakeup from PB13 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 4 **WUF16**: PB12 I/O wakeup flag.
0: no wakeup from PB12 I/O occurred since last clear.
1: a wakeup from PB12 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 3 **WUF15**: PA3 I/O wakeup flag.
0: no wakeup from PA3 I/O occurred since last clear.
1: a wakeup from PA3 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 2 **WUF14**: PA2 I/O wakeup flag.
0: no wakeup from PA2 I/O occurred since last clear.
1: a wakeup from PA2 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 1 **WUF13**: PA1 I/O wakeup flag.
0: no wakeup from PA1 I/O occurred since last clear.
1: a wakeup from PA1 I/O occurred since last clear.
Cleared by writing 1 in this bit.

Bit 0 **WUF12**: PA0 I/O wakeup flag.
0: no wakeup from PA0 I/O occurred since last clear.
1: a wakeup from PA0 I/O occurred since last clear.
Cleared by writing 1 in this bit.

5.7.15 Shutdown I/O Wakeup Enable register (PWRC_SDWN_WUEN)

*A*ddress offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **WUEN**: PB0 I/O wakeup from shutdown enable.
0: PB0 pin wakeup from shutdown disable.
1: PB0 pin wakeup from shutdown enable.

5.7.16 Shutdown I/O Wakeup Polarity register (PWRC_SDWN_WUPOL)

*A*ddress offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPOL
															rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 0 **WUPOL**: PB0 I/O wakeup from shutdown polarity.

0: PB0 pin wakeup from shutdown on high pulse or high level detection.

1: PB0 pin wakeup from shutdown on low pulse or low level detection.

5.7.17 Shutdown I/O Wakeup Flag register (PWRC_SDWN_WUF)

This register is reset on PORESETn.

*A*ddress offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF
															rc_w0

Bits 31:16 Reserved, must be kept at reset value.

Bit 0 **WUF**: PB0 I/O wakeup from shutdown flag.

0: shutdown wakeup from PB0 pin not occurred.

1: shutdown wakeup from PB0 pin occurred.

5.7.18 Debug register (PWRC_DBGR)

This register is used for debug featu

*A*ddress offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIS_PRECH[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEEPSTOP2
rw	rw	rw													rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **DIS_PRECH[2:0]**: disable SMPS PRECHARGE during Deepstop (debug only).
111: PRECHARGE and SMPS monitoring disabled
101: PRECHARGE enabled only at Deepstop exit (PWRC_CR5.SMPSLPOPEN must be set).
Other values: Deepstop PRECHARGE enabled

Bits 12:1 Reserved, must be kept at reset value.

Bit 0 **DEEPSTOP2**: Deepstop2 low power saving emulation enable.
0: normal Deepstop is applied
1: Deepstop2 (debugger features not lost) is applied instead of Deepstop.

5.7.19 Extended status and reset register (PWRC_EXTSRR)

This register provides flags about Bluetooth activity start and Deepstop sequence occurrence or not.

*A*ddress offset: 0x88

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RFPHASEF	DEEPSTOPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rc_w1	rc_w1									

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RFPHASEF**: RFPHASE Flag.

0: the Bluetooth LE IP does not require any attention.

1: the Bluetooth LE IP is awake and may require a system attention.

This bit is set by hardware when a radio wakeup event occurs.

This bit is reset by hardware when the Bluetooth LE IP raises the “ready to sleep” information.

The software can reset this bit by writing 1 in it.

Bit 9 **DEEPSTOPF**: System DEEPTSTOP Flag.

0: the device did not enter Deepstop mode.

1: the device entered a Deepstop mode.

This bit is set by hardware when a Deepstop sequence occurred.

The software can reset this bit by writing 1 in it.

Bits 8:0 Reserved, must be kept at reset value.

5.7.20 Trimming values register (PWRC_TRIMR)

This register provides the trimming values applied by hardware.

*A*ddress offset: 0x90

Reset value: 0x---- ---- (0x0000 0307 when device is not trimmed)

Note: A Hardware default value is defined after reset. It is overwritten by the information stored in flash memory on trimmed devices, and kept as-is for devices without trimming information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SMPS_TRIM[2:0]			TRIM_MR[3:0]				RFD_REG_TRIM[3:0][			
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **SMPS_TRIM[2:0]**: SMPS output voltage trimming.
Note: the default value is the centered value (“011”).

Bits 7:4 **TRIM_MR[3:0]**: Main regulator voltage trimming.
Default value when no trimming is “0000”.

Bits 3:0 **RFD_REG_TRIM[3:0]**: RF LDO trimming.
Default value when no trimming is “0111”.

5.7.21 Software trimming values register (PWRC_ENGTRIM)

This register allows the software to overwrite the hardware trimming values.

*A*ddress offset: 0x94

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SMPS_TRIM[2:0]			SMPSTRIMEN	TRIM_MR[3:0]				TRIMMREN	RFD_REG_TRIM[3:0][				TRIMRFDREGEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits13:11 **SMPS_TRIM[2:0]**: SMPS output voltage trimming chosen by the software.

Bit 10 **SMPSTRIMEN**: SMPS_TRIM software overwrite enable.
0: Hardware trimming value from TRIMR.SMPS_TRIM bit field is applied.
1: Trimming value from ENGTRIM.SMPS_TRIM bit field is applied

Bits 9:6 **TRIM_MR[3:0]**: Main regulator voltage trimming chosen by the software.

Bit 5 **TRIMMREN**: TRIM_MR software overwrite enable.
0: Hardware trimming value from TRIMR.TRIM_MR bit field is applied.
1: Software trimming value from ENGTRIM.TRIM_MR bit field is applied

Bits 4:1 **RFD_REG_TRIM[3:0]**: RF LDO trimming chosen by the software.

Bit 0 **TRIMRFDREGEN**: RFD_REG_TRIM software overwrite enable.
0: Hardware trimming value from TRIMR.RFD_REG_TRIM bit field is applied.
1: Software trimming value from ENGTRIM.RFD_REG_TRIM bit field is applied

5.7.22 PWRC register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the PWRC base address location in the STM32WB09xE.

Note: All the PWRC registers are retained during Deepstop mode.

The shaded cells indicate the bit fields located in the VDD33 power domain. This implies the associated feature is applied even during Shutdown state (but are lost at Shutdown mode exit as a PORESETn is generated).

Table 12. PWRC register map and reset values

Offset	Register	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
0x00	**PWRC_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APC	IBIAS_RUN_STATE	IBIAS_RUN_AUTO	ENSDNBOR	LPMS
	Reset value																												1	0	1	0	0
0x04	**PWRC_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIORET	RAMRET3	RAMRET2	RAMRET1	DBGRET	PVDLS			PVDE
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0
0x08	**PWRC_CR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EIWL	EIWL2	EWBLEHCPU	EWBLE	EWU11	EWU10	EWU9	EWU8	EWU7	EWU6	EWU5	EWU4	EWU3	EWU2	EWU1	EWU0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**PWRC_CR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUP11	WUP10	WUP9	WUP8	WUP7	WUP6	WUP5	WUP4	WUP3	WUP2	WUP1	WUP0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**PWRC_SR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWUF	IWUF2	WBLEHCPUF	WBLEF	WUF11	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**PWRC_SR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOBOOTVAL[3:0]				PVDO	Res.	Res.	REGLPS	IOBOOTVAL2[3:0]				Res.	SMPSRDY	SMPSENR	SMPSBYPR
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	1	1	-	-	-	-	-	1	1	0
0x18	Reserved																																

Table 12. PWRC register map and reset values (continued)

Offset	Register	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
0x1C	**PWRC_CR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPS_PRECH_CUR_SEL[1:0]		CLKDETR_DISABLE	SMPS_ENA_DCM	NOSMPS	SMPSFBYP	SMPSLPOEN	Res.	Res.	SMPSBOMSEL[1:0]		SMPSLVL[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	0	0	0	0	0		-	0	1	0	1	0	0
0x20	**PWRC_PUCRA**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUA11	PUA10	PUA9	PUA8	Res.	Res.	Res.	Res.	PUA3	PUA2	PUA1	PUA0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	0	1	1	1
0x24	**PWRC_PDCRA**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDA11	PDA10	PDA9	PDA8	Res.	Res.	Res.	Res.	PDA3	PDA2	PDA1	PDA0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	1	0	0	0
0x28	**PWRC_PUCRB**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUB15	PUB14	PUB13	PUB12	Res.	Res.	Res.	Res.	PUB7	PUB6	PUB5	PUB4	PUB3	PUB2	PUB1	PUB0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	1	1	1	1
0x2C	**PWRC_PDCRB**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDB15	PDB14	PDB13	PDB12	Res.	Res.	Res.	Res.	PDB7	PDB6	PDB5	PDB4	PDB3	PDB2	PDB1	PDB0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0
0x30	**PWRC_CR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU19	EWU18	EWU17	EWU16	EWU15	EWU14	EWU13	EWU12
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x34	**PWRC_CR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUP19	WUP18	WUP17	WUP16	WUP15	WUP14	WUP13	WUP12
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x38	**PWRC_SR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF19	WUF18	WUF17	WUF16	WUF15	WUF14	WUF13	WUF12
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x3C	**PWRC_SDWN_WUEN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUEN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0x40	**PWRC_SDWN_WUPOL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPOL
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0x44	**PWRC_SDWN_WUF**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0x48 - 0x80	Reserved																																
0x84	**PWRC_DBGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_PRE[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEEPSTOP2
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	-	-	-	-	-	-	-	-	-	-	-	0

Table 12. PWRC register map and reset values (continued)

Offset	Register	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
0x88	**PWRC_EXTSRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFPHASEF	DEEPSTOPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-
0x90	**PWRC_TRIMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPS_TRIM[3:0]			TRIM_MR[3:0]				RFD_REG_TRIM[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1	0	0	0	0	0	1	1	1
0x94	**PWRC_ENGTRIM**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPS_TRIM[3:0]			SMPSTRIMEN	TRIM_MR[3:0]				TRIMMREN	RFD_REG_TRIM[3:0]				TRIMRFDREGEN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5.8 Programmer's model

5.8.1 Reset reason management

The CPU has many reasons to be reset and executes its reset handler. *Table 13* provides an overview of the flags that can help the embedded software to get the root cause of the CPU restart.

Table 13. Flags versus CPU reboot reason

CPU reboot reason	RCC_CSR					PWRC_ISCR (in SYSCFG)	PWRC_EXTSRR
	LOCKUPRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	WAKEUP_ISC	DEEPSTOPF
POR/BOR reset	-	-	-	1	1	-	-
NRSTn pad reset	-	-	-	-	1	-	-
Watchdog reset	-	1	-	-	1	-	-
System reset (CPU request)	-	-	1	-	1	-	-
LOCKUP reset	1	-	-	-	1	-	-
Deepstop exit on wakeup event	-	-	-	-	-	1	1
Deepstop exit on watchdog reset	-	1	-	-	1	-	-
Deepstop exit on NRSTn pad reset	-	-	-	-	1	-	-
Deepstop exit on POR/BOR	-	-	-	1	1	-	-
Shutdown exit	-	-	-	1	1	-	-

If the reboot reason is a wakeup from Deepstop, then the wakeup source(s) can be read in the PWRC_SR1 register as shown in *Table 14*.

Table 14. Wakeup reason flags

Wakeup reason	IWUF2	PWRC_SR1			
		IWUF	WBLEHCPUF	WBLEF	WUFx
Wakeup on Bluetooth LE event	-	-	-	1	-
Wakeup on Host timer in MR_BLE event	-	-	1	-	-
Wakeup on RTC event	-	1	-	-	-
Wakeup on LPUART event	1	-	-	-	-
Wakeup on I/Os	-	-	-	-	1

Note: If several (enabled) wakeup events occur, several bits are high in the PWRC_SR1.

The wakeup flags are set as soon as a wakeup event (enabled in PWRC_CR3 register) occurs, the associated flag is set in the PWRC_SR1 register even if the device is in active mode or in the sleep exit sequence (initiated by another wakeup source).

Caution: Those flags have to be cleared by software knowing a Deepstop entry sequence cannot happen if a wakeup flag is already active when the system request a Deepstop mode.

5.8.2 SMPS output level re-programming

The SMPS output voltage cannot be modified on-the-fly why the SMPS is in use of more than one step. When the software needs to re-program the SMPS output voltage to another value, the following sequence must be respected:

- Set PWRC_CR5.SMPSBYP = 1
- Wait for PWRC_SR2.SMPSRDY = 0
- Program the new targeted value in PWRC_CR5.SMPSLVL[3:0]
- Clear PWRC_CR5.SMPSBYP = 0
- Wait for PWRC_SR2.SMPSRDY = 1

Caution: This sequence must be launched only when no radio activity / RF transfer is ongoing.

6 Reset and clock controller (RCC)

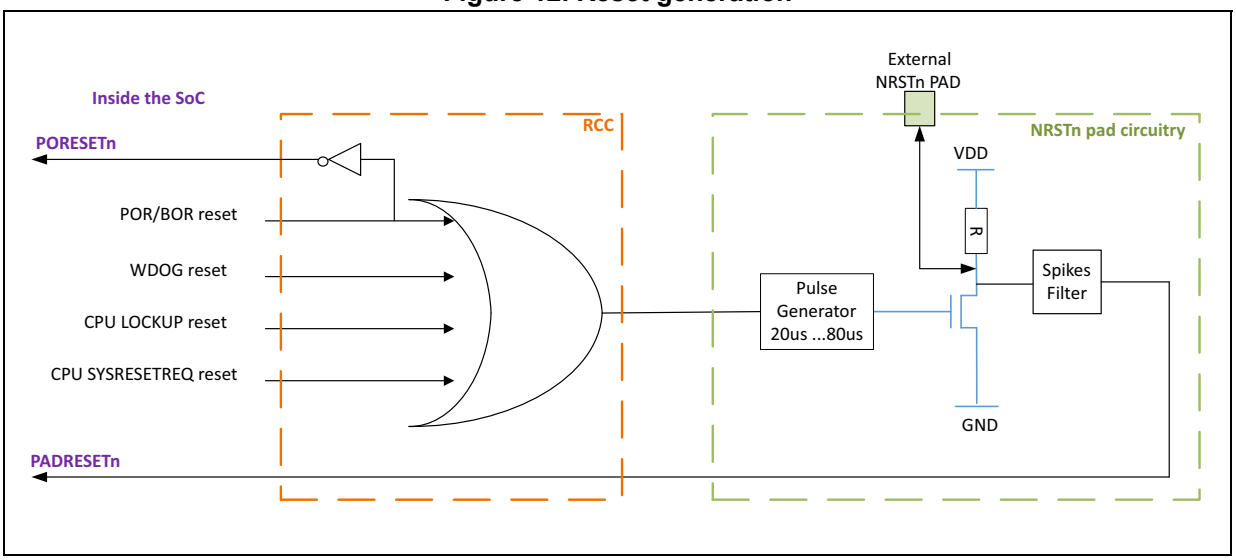
The RCC block manages the clock and reset generation for all the peripherals of the STM32WB09xE device.

6.1 Reset management

6.1.1 General description

Figure 12 shows the general principle of reset generation.

Figure 12. Reset generation



Note: The system reset information is output on the NRSTn pad to inform the external world and reset other elements on the board if needed.

Two different resets are available in the design:

- PORESETn: this reset is provided by the APMU analog block and corresponds to a POR or BOR root cause. It is linked to power voltage ramp-up or ramp-down.

The PORESETn reset impacts all the resources of the device.

Note: A Shutdown exits is equivalent to a POR/BOR situation and generates a PORESETn.

- PADRESETn (also called system reset): this reset is built through several sources:
  - PORESETn,
  - the watchdog reset,
  - the CPU LOCKUP reset,
  - the CPU software system reset,
  - the NRSTn external pad.

Note: The system reset is called PADRESETn as when an internal reset source is activated (watchdog, software, and so on), the NRSTn pad toggles to inform the external world a reset occurs.

This system reset resets all the resources of the device except:

- debug features (SWD, test registers...)
- flash controller key management part
- RTC timer
- power controller (PWRC)
- part of the RCC registers

The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of reset from the NRSTn external pad, the reset pulse is generated when the pad is asserted low.

6.1.2 Power reset

The PORESETn signal is active when the power supply of the device is below a threshold value or when the regulator does not provide the target voltage.

The PORESETn resets all the resources of the device

6.1.3 Watchdog reset

The STM32WB09xE device embeds a watchdog timer which may be used to recover from software crashes.

See *Section 20: Independent watchdog (IWDG)* for details about watchdog usage and programming.

6.1.4 LOCKUP reset

The Cortex-M0+ generates a LOCKUP to indicate the core is in the lock-up state resulting from an unrecoverable exception.

The LOCKUP reset is masked if a debugger is connected to the Cortex-M0+. The user can use the SWD to reset or recover the code in this case.

6.1.5 System reset request

The system reset request is generated by the debug circuitry of the Cortex-M0+. The debugger sets the SYSRESETREQ bit of the Application Interrupt and Reset control Register (AIRCR). This system reset request through the AIRCR can also be done by the embedded software (in hard fault handler for instance).

For more details on the Cortex-M0+ system control and ID registers, refer to section B3.2.2 of the ARMv6-M Architecture Reference Manual *[4]*.

6.1.6 Deepstop exit

The low power Deepstop state leads to switch off a part of the 1.2V (power domain called V12i), while keeping the rest of the 1.2V at 1V (power domain called V12o) and the 3.3V (VDDIO).

When the device exits the Deepstop mode, only the V12i power domain is reset as it is the only power domain that lost the power supply.

6.2 Clock management

Three different clock sources may be used to drive the system clock (CLK_SYS) in the STM32WB09xE:

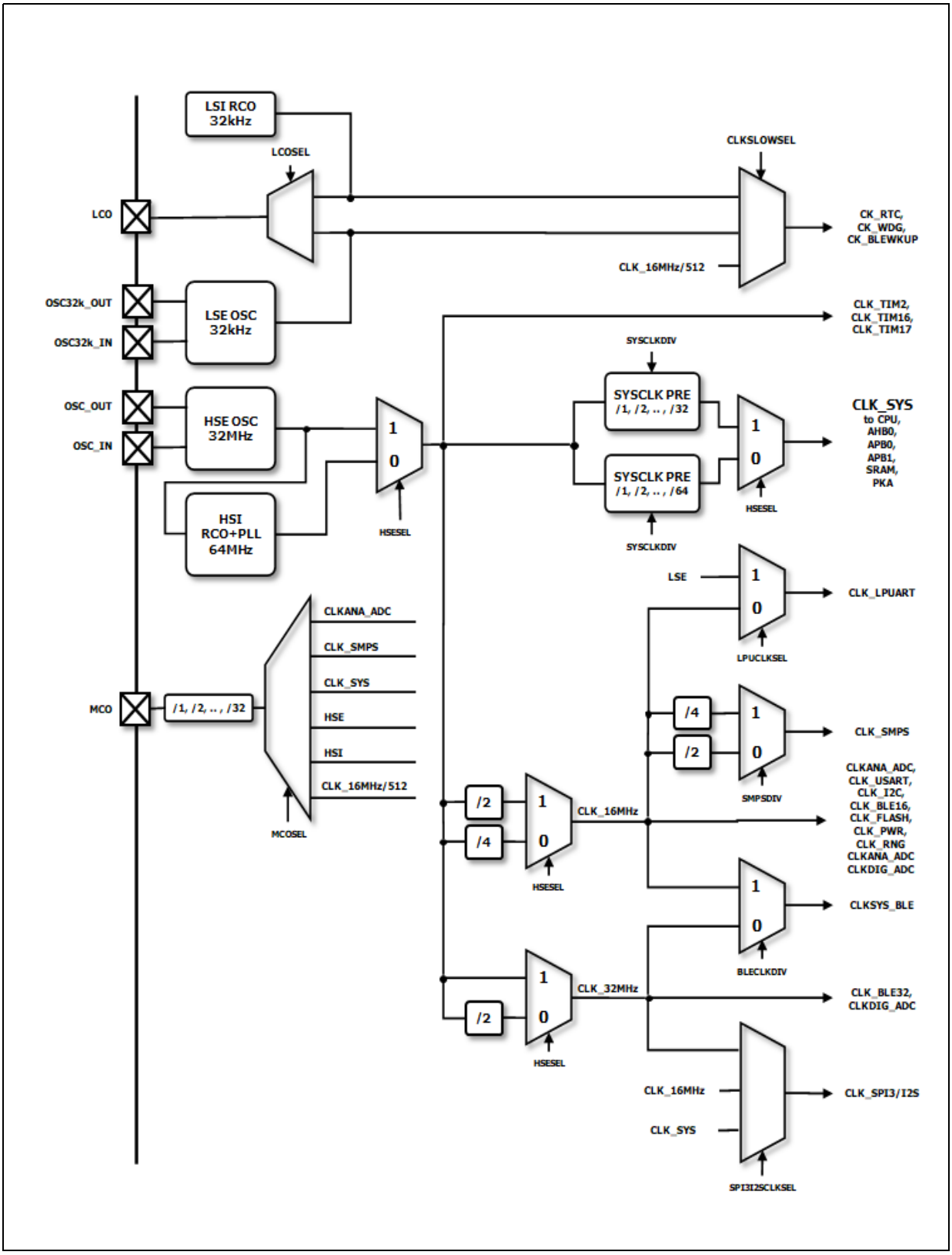
- HSI: high speed internal 64 MHz RC oscillator (provided by the RC64MPLL analog block),
- PLL64M: 64 MHz PLL clock (provied by the RC64MPLL analog block),
- HSE (high speed external):
  - high speed 32 MHz external crystal.
  - or provided by a single ended 32 MHz input instead of a crystal.

The STM32WB09xE device has also a slow frequency clock tree used by some peripherals (RTC, watchdog, LPUART and MR_BLE radio timer). Three different clock sources can be used for this slow clock tree:

- LSI: low speed low drift internal RC with a fixed frequency between 24 kHz and 49 kHz depending on the sample. It is called “32 kHz” clock inside this document to simplify.
- LSE:
  - 32.768 kHz low speed external crystal.
  - or provided by a single-ended 32.768 kHz input instead of a crystal.
- The always 16 MHz clock divided by 512 (not available for LPUART). In this case, the slow clock is not available in Deepstop low power mode.

Figure 13 provides an overview of the clock tree in the STM32WB09xE.

Figure 13. Clock tree generation



6.2.1 System clock details

The HSI and the PLL64M clocks are provided by the same analog block called RC64MPLL. The 64 MHz clock output by this block can be:

- A non accurate clock (target is 1% typical) when no external XO provides an input clock to this block.
- An accurate clock when the external XO provides the 32 MHz and once its internal PLL is locked.

Note: The usage of PLL64M or HSE as clock source is mandatory for Bluetooth radio operations (need of a high accuracy on the clock).

The software process to switch the system on the accurate clock is indicated in *Section 6.7: Programmer's model*.

This fast clock source is used to generate all the fast clock of the device through dividers as shown in *Figure 13*.

After reset, the CLK_SYS is divided by four to provide a 16 MHz to the whole system (CPU, DMA, memories and peripherals).

Then the software can program another system clock frequency in the following list:

- 1 MHz (forbidden when radio or ADC is in use)
- 2 MHz (forbidden when radio or ADC is in use)
- 4 MHz (forbidden when radio or ADC is in use)
- 8 MHz (forbidden when radio is in use)
- 16 MHz
- 32 MHz
- 64 MHz

Note: Forbidden configuration means that the "in use" feature cannot work if the system clock runs at this frequency.

Note: Special care must be taken when programming the CLK_SYS as some constraints need to be respected:

CLK_SYS frequency must be greater or equal to CLK_SYS_BLE.

6.2.2 Peripherals clock details

This fast clock source is also used to generate several internal fast clocks in the system:

- A TIM2, TIM16/17 kernel clock that is the maximum reachable frequency of the system (64 MHz in RC64MPLL configuration and 32 MHz in HSE or configurations).
- An always 32 MHz requested by few peripherals like the MR_BLE radio IP for instance
- An always 16 MHz requested by few peripherals like serial interfaces (to maintain fixed baud rate while system clock is switching from a frequency to another), ADC or like flash controller and MR_BLE radio IP (to have a fixed reference clock to manage delays).

Most of the peripherals use only the system clock (CLK_SYS) except:

- I2C, USART:
  - In parallel of the system clock, they use an always 16 MHz clock to have a fixed reference clock for baud rate management. The goal is to allow the CPU to boost

or slow down the system clock (depending on on-going activities) without impacting a potential on-going serial interface transfer on external I/Os.

- LPUART:
  - In parallel of the system clock, it uses an always 16 MHz clock to have a fixed reference clock for baud rate management and LSE clock when active in Deepstop. The goal is to allow the CPU to boost or slow down the system clock (depending on on-going activities) without impacting a potential on-going serial interface transfer on external I/Os.
- SPI:
  - When using the I2S mode, the baud rate is managed through the always 16 MHz, always 32 MHz clock or system clock (SYS_CLK).

Note: The CPU/system clock frequency must be equal or slower than the I2S clock frequency.

  - When running in other modes than the I2S, the baud rate is managed by the system clock This implies there baud rate is impacted by dynamic system clock frequency changes.
- TRNG:
  - In parallel of the system clock, the TRNG uses an always 16 MHz clock to generate at a constant frequency the random number whatever the system clock frequency. To use the TRNG peripheral the system clock frequency must be at least 32 MHz.
- Flash controller:
  - In parallel of the system clock, the flash controller uses an always 16 MHz clock to generate specific delays required by the flash memory during programming and erase operation for instance.
- MR_BLE IP
  - MR_BLE IP does not use directly the system clock for its APB / AHB interfaces but the system clock with a potential divider (1 or 2 or 4). *Table 15* shows the supported configurations.
  - In parallel of the CLK_SYS_BLE, the MR_BLE uses an always 16 MHz and an always 32 MHz for modulator, demodulator and to have a fixed reference clock to manage specific delays.

Table 15. CPU versus MR_BLE clock dependency

CLK_SYS	CLK_SYS_BLE
1 MHz / 2 MHz / 4 MHz / 8 MHz	Not possible to use MR_BLE IP
16 MHz	16 MHz (CLKBLEDIV = 4)
32 MHz	16 MHz (CLKBLEDIV = 4) or 32 MHz (CLKBLEDIV = 2)
64 MHz	16 MHz (CLKBLEDIV = 4) or 32 MHz (CLKBLEDIV = 2)

- ADC:
  - In parallel of the system clock, the ADC uses a 64 MHz prescaled clock (called CLKANA_ADC and CLKDIG_ADC) running at 16 MHz.

Note: When the ADC is used, the system clock must run at minimum 8 Mhz to be able to read the ADC sample before they are overloaded by a new sample.

Note: To avoid SNR degradation of the ADC, SMPS and ADC clocks must be synchronous.

6.2.3 Slow clock frequency details

As explained at the beginning of the clock management sub-section, three different clock sources can be used for this slow clock tree:

- LSI: low speed low drift internal RC with a fixed frequency between 24 kHz and 49 kHz depending on the sample. It is called “32 kHz” clock in this document to simplify.
- LSE: 32.768 kHz low speed external crystal (or single-ended input frequency).

Note: If the external oscillator is used, the PB12/PB13 I/Os is automatically connected to this feature when RCC_CR.LSEON bit is set (GPIO_MODERX configuration is overloaded).

Caution: If APC bit is set, the user has to disable the PUB12/PUB13/PDB12/PDB13 bits on PB12/PB13 by software *I/O Port B pull-up control register (PWRC_PUCRB)* to have the feature working fine. otherwise if APC is reset, the pull-up, pull-down of PB12/13 are automatically configured.

- An always 32 kHz clock equal to HSI/2048 if HSESEL=’0’ else HSE/1024. In this case, the slow clock is not available in Deepstop low power mode.

Only one source at a time drives the whole low speed clock tree.

Note: By default after a PORESETn, all low speed sources are OFF. After a PADRESETn, the slow clock configuration is the one programmed before the PADRESETn.

The slow clock activation and selection stays relevant during the Deepstop low power mode and at wakeup as it clocks the timers involved in wakeup events generation.

Note: If LSI configuration is used, the software must measure the slow clock frequency to know the associated period that is used by the timers. A slow clock measurement feature is available in the MR_BLE IP.

6.3 System frequency switch while MR_BLE is used

The CPU/system clock frequency can be from 1 MHz to 64 MHz while the MR_BLE clock frequency can be 16 MHz or 32 MHz.

When the radio is used on the device, the system clock frequency selection must respect some rules:

- The system clock frequency must be 16 MHz, 32 MHz or 64 MHz and greater than or equal to the MR_BLE frequency. Other options make the radio nonfunctional.
- Changing the frequency of the system must be done through the RCC_CSCMDR register mechanism to avoid any risk to crash the radio scenarios.

This proper system frequency switch is managed through the collaboration of several blocks:

- the RCC (see *Section 6.6.6: Clock switch command register (RCC_CSCMDR)*)
- the AHBUPCONV and the AHBDOWNCONV blocks (see *Section 3: AHB up/down converter*)

Using this safe mechanism, the software requests a system clock frequency change and is informed by the hardware when the new frequency is really in place through a status bit (see

Section 6.6.6: Clock switch command register (RCC_CSCMDR)) and an associated interrupt line on the CPU (see *Section 2.3.2: Interrupts*).

The software sequence is described in the *Section 6.7.3: Changing the system clock frequency while the MR_BLE is enabled* sub-section.

6.4 Clock observation on external pad

It is possible to output some internal clocks on external pads:

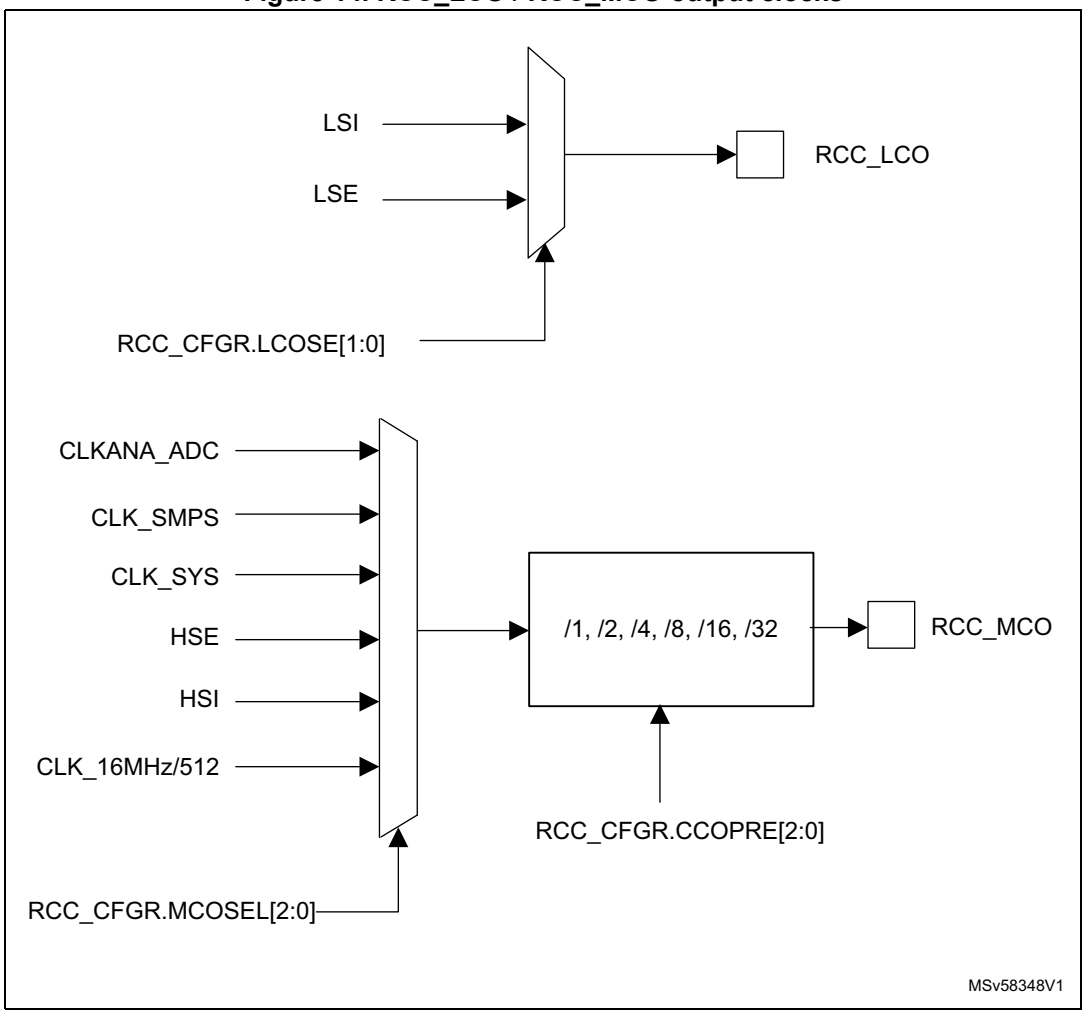
- the low speed clocks can be output on the RCC_LCO I/O,
- the high speed clocks can be output on the RCC_MCO I/O.

This is possible by programming the associated I/O in the right alternate function (see *Table 6: Interrupt vectors* and *Table 8: Alternate modes 3, 4, and 6*).

The selection of the clock to output for each I/O is programmable through a RCC register (see *Section 6.6.2: Clock configuration register (RCC_CFGR)* for more details).

Figure 14 shows the possible configurations to output an internal clock.

Figure 14. RCC_LCO / RCC_MCO output clocks



6.5 Miscellaneous

6.5.1 IO BOOSTER

Some analog switches are used to select the analog VINM/P pair input signals to be used by the ADC. An IO BOOSTER block has been added to boost the voltage on the command of those analog switches when the VBAT goes below a threshold (2.7V) to guarantee the good behavior of those switches. This block has to be enabled by the software when needed through RCC_CFGR.IOBOOSTEN bit.

6.6 RCC registers

6.6.1 Clock sources control register (RCC_CR)

This register controls the enable on the different clock sources (low and high speed).

Note: The control bits linked to high speed clock source are reset on PADRESETn. The control bits linked to slow speed clock source are reset on PORESETn only (identified by the table footnote).

As this register is in V12o power domain, its content is not modified after a wakeup from Deepstop and system clock is restored with configuration present before Deepstop mode entry.

Address offset: 0x00

Reset value: 0x0000 1400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDY	HSEON
														r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HSIPLLRDY	HSIPLLON	HSEPLLBUFON	Res.	HSIRDY	LOCKDET_NSTOP			LSEBYP	LSERDY	LSEON	LSIRDY	LSION	Res.	Res.
	r	rw	rw		r	rw	rw	rw	rw	r	rw	r	rw		

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **HSERDY**: External High Speed clock flag.
This bit is set by hardware to indicate that HSE oscillator (32 MHz XO) is stable.
– 0: HSE oscillator is not ready.
– 1: HSE oscillator is ready.

Bit 16 **HSEON**: External High Speed clock enable.
The software has to set the bit to start the XO 32 MHz and clear the bit to stop it.
– 0: HSE oscillator is OFF.
– 1: HSE oscillator is ON.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **HSIPLLRDY**: Internal High Speed clock PLL flag.
This bit is set by hardware to indicate that the RC64MPLL pll is locked.
– 0: RC64MPLL pll is unlocked.
– 1: RC64MPLL pll is locked.

Bit 13 **HSIPLLON**: Internal High Speed clock PLL enable.
The software has to set the bit to request a RC64MPLL lock on HSE and clear the bit to stop it.
– 0: RC64MPLL pll if OFF.
– 1: RC64MPLL pll is ON.

Bit 12 **HSEPLLBUFON**: External high speed clock buffer for PLL RF2G4 enable.
The software has to set the bit when the radio is used (to have the 2.4 GHz PLL working).
– 0: HSE pll RF2G4 buffer is OFF.
– 1: HSE pll RF2G4 buffer is ON.
Warning: this bit must be set when the radio is used. The only reason to clear this bit would be to reduce power consumption for application not using the radio on this device.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **HSIRDY**: Internal High Speed clock flag.
This bit is set by hardware to indicate that internal 64 MHz RC is stable.
– 0: internal 64 MHz RC is not ready.
– 1: internal 64 MHz RC is ready.

Bits 9:7 **LOCKDET_NSTOP**: define a time window target for the counter of the lock detector block in charge to manage the HSIPLLRDY information (PLL indicated as locked if the analog lock signal stays high and stable during this time window).
The formula to define the time window target is the following:
time window target = (LOCKDET_NSTOP + 1) x 64.

Bit 6(1) (2) **LSEBYP**: External Low Speed clock bypass.
This bit needs to be set when the slow clock is directly provided through RCC_OSC32_IN pin.
– 0: No LSE oscillator bypass.
– 1: LSE oscillator bypass is enabled.

Bit 5 (1) **LSERDY**: External Low Speed clock flag.
This bit is set by hardware to indicate that the slow clock has started.
– 0: LSE oscillator is not ready.
– 1: LSE oscillator is ready.
Note: this status bit is true whatever the chosen configuration (external 32 kHz oscillator -= LSEON or external clock provided on RCC_OSC32_IN = LSEBYP).

Bit 4(1) (2) **LSEON**: External Low Speed clock enable.
The software has to set the bit to start the XO 32 kHz and clear the bit to stop it.
– 0: LSE oscillator is OFF.
– 1: LSE oscillator is ON.

Bit 3(1) **LSIRDY**: Internal Low Speed clock flag.
This bit is set by hardware to indicate that internal low speed RC is stable.
– 0: internal low speed RC is not ready.
– 1: internal low speed RC is ready.

Bit 2(1) **LSION**: Internal low speed RC clock enable.
The software has to set the bit to start the internal slow clock RO and clear the bit to stop it.
– 0: LSI RC is OFF.
– 1: LSI RC is ON.

Bits 1:0 Reserved, must be kept at reset value.

1. This bit is reset on PORESETn only.
2. The LSEBYP and LSEON bits must not be used at the same time. If the user decides to dynamically change the slow clock source between external XO and clock injection on RCC_OSC32_IN, he has to ensure both LSEON and LSEBYP are low at a time to reset the LSERDY flag.

6.6.2 Clock configuration register (RCC_CFGR)

Note: The control bits linked to high speed clock source are reset on PADRESETn. The control bits linked to slow speed clock source are reset on PORESETn only (identified by the table footnote).

Address offset: 0x08

Reset value: 0x0000 0240

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCOPRE[2:0]			MCOSEL[2:0]			LCOSEL[1:0]		SPI3I2SCLKSEL [1:0]		Res.	Res.	LCOEN	IOBOOSTCLKEN	IOBOOSTEN	CLKSLOWSEL[1]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLKSLOWSEL[0]	Res.	LPUCLKSEL	SMPSDIV	Res.	CLKSYSDIV_STATUS[2:0]			CLKSYSDIV[2:0]			Res.	HSESEL_STATUS	STOPHSI	HSESEL	Res.
rw		rw	rw		r	r	r	rw	rw	rw		r	rw	rw	

Bits 31:29 **CCOPRE**: Configurable Clock Output Prescaler.
- 000: CCO clock is divided by 1
- 001: CCO clock is divided by 2
- 010: CCO clock is divided by 4
- 011: CCO clock is divided by 8
- 100: CCO clock is divided by 16
- 101:CCO clock is divided by 32
- others: reserved

Note: Glitches propagation possible if CCOPRE[2:0] value is modified while RCC_MCO output is enabled on the IO.

Bits 28:26 **MCOSEL**: Main Configurable Clock Output Selection.
- 000: RCC_MCO output disabled. No clock on RCC_MCO pad.
- 001: system clock
- 010: reserved
- 011: HSI_64M = RC64MPLL block output clock (can be internal 64 MHz or PLL 64 MHz accuracy).
- 100: HSE (external 32 MHz oscillator)
- 101: always16 MHz clock divided by 512
- 110: SMPS clock
- 111: ADC clock

Note: Glitches propagation possible if MCOSEL[2:0] value is modified while RCC_MCO output is enabled on the IO.

Bits 25:24(1) **LCOSEL**: Low speed Configurable Clock Output Selection.
- 00: RCC_LCO output disabled. No clock on RCC_LCO pad.
- 01: not used
- 10: LSI (internal slow clock RC) clock
- 11: LSE (external 32 kHz)

Note: Glitches propagation possible if LCOSEL[1:0] value is modified while RCC_LCO output is enabled on the IO.

Bits 23: 22 **SPI3I2SCLKSEL**[1:0]: Selection of I2S clock for SPI3 IP.
- 00: 16 MHz peripheral clock (default)
- 01: 32 MHz peripheral clock
- 1x: 64 MHz peripheral clock (available only when HSESEL=0)

Note: the I2S clock frequency must be higher or equal to the system clock (configured through RCC_CFGR.CLKSYSDIV[2:0] bit field).

Bits 21:20 Reserved, must be kept at reset value.

Bits 19(1) **LCOEN**: RCC_LCO enable on PA10 also in Deepstop.
- 0: RCC_LCO output on PA10 is disabled
- 1: RCC_LCO output on PA10 is enabled.

Bit 18 **IOBOOSTCKEN**: IO BOOSTER clock enable (see *Section 6.5.1: IO BOOSTER* for details).
- 0: IO BOOSTER block doesn't use RCC clock
- 1: IO BOOSTER block uses RCC clock.

Bit 17 **IOBOOSTEN**: IO BOOSTER enable (see *Section 6.5.1: IO BOOSTER* for details).
- 0: IO BOOSTER block is disabled
- 1: IO BOOSTER block is enabled.

Bits 16:15(1) **CLKSLOWSEL**: low speed clock source selection.
- 00: not used
- 01: LSE (external oscillator). This source can be kept during Deepstop mode.
- 10: LSI (internal RC). This source can be kept during Deepstop mode.
- 11: always 16 MHz divided by 512

Note: No glitch mechanism has been added so glitches may appear on slow clock when the user changes its source.

Bits 14 Reserved, must be kept at reset value.

Bit 13(1) **LPUCLKSEL**: Selection of LPUART clock
- 0: 16 MHz peripheral clock (default)
- 1: LSE clock

Bit 12 **SMPSDIV**: SMPS clock prescaling factor.
- 0: SMPS clock is 8 MHz
- 1: SMPS clock is 4 MHz

Bits 11 Reserved, must be kept at reset value.

Bits 10:8 **CLKSYSDIV_STATUS**: system clock frequency status

Set and cleared by hardware to indicate the actual system clock frequency. This register must be read to be sure that the new frequency, selected by CLKSYSDIV, has been applied.
- 000: system clock frequency is 64 MHz
- 001: system clock frequency is 32 MHz
- 010: system clock frequency is 16 MHz
- 011: system clock frequency is 8 MHz
- 100: system clock frequency is 4 MHz
- 101: system clock frequency is 2 MHz
- 110: system clock frequency is 1 MHz
- 111: not used.

The actual clock frequency switching can be delayed of up to 128 system clock cycles, depending on the RCC internal counter status at the moment the new CLKSYSDIV is applied

Bits 23: 22 **SPI3I2SCLKSEL**[1:0]: Selection of I2S clock for SPI3 IP.
- 00: 16 MHz peripheral clock (default)
- 01: 32 MHz peripheral clock
- 1x: 64 MHz peripheral clock (available only when HSESEL=0)

Note: the I2S clock frequency must be higher or equal to the system clock (configured through RCC_CFGR.CLKSYSDIV[2:0] bit field).

Bits 21:20 Reserved, must be kept at reset value.

Bits 19(1) **LCOEN**: RCC_LCO enable on PA10 also in Deepstop.
- 0: RCC_LCO output on PA10 is disabled
- 1: RCC_LCO output on PA10 is enabled.

Bit 18 **IOBOOSTCKEN**: IO BOOSTER clock enable (see *Section 6.5.1: IO BOOSTER* for details).
- 0: IO BOOSTER block doesn't use RCC clock
- 1: IO BOOSTER block uses RCC clock.

Bit 17 **IOBOOSTEN**: IO BOOSTER enable (see *Section 6.5.1: IO BOOSTER* for details).
- 0: IO BOOSTER block is disabled
- 1: IO BOOSTER block is enabled.

Bits 16:15(1) **CLKSLOWSEL**: low speed clock source selection.
- 00: not used
- 01: LSE (external oscillator). This source can be kept during Deepstop mode.
- 10: LSI (internal RC). This source can be kept during Deepstop mode.
- 11: always 16 MHz divided by 512

Note: No glitch mechanism has been added so glitches may appear on slow clock when the user changes its source.

Bits 14 Reserved, must be kept at reset value.

Bit 13(1) **LPUCLKSEL**: Selection of LPUART clock
- 0: 16 MHz peripheral clock (default)
- 1: LSE clock

Bit 12 **SMPSDIV**: SMPS clock prescaling factor.
- 0: SMPS clock is 8 MHz
- 1: SMPS clock is 4 MHz

Bits 11 Reserved, must be kept at reset value.

Bits 10:8 **CLKSYSDIV_STATUS**: system clock frequency status

Set and cleared by hardware to indicate the actual system clock frequency. This register must be read to be sure that the new frequency, selected by CLKSYSDIV, has been applied.
- 000: system clock frequency is 64 MHz
- 001: system clock frequency is 32 MHz
- 010: system clock frequency is 16 MHz
- 011: system clock frequency is 8 MHz
- 100: system clock frequency is 4 MHz
- 101: system clock frequency is 2 MHz
- 110: system clock frequency is 1 MHz
- 111: not used.

The actual clock frequency switching can be delayed of up to 128 system clock cycles, depending on the RCC internal counter status at the moment the new CLKSYSDIV is applied

Bits 7:5 **CLKSYSDIV**: system clock divided factor from HSI_64M.
- 000: system clock frequency is 64 MHz (**not available when HSESEL=1**)
- 001: system clock frequency is 32 MHz
- 010: system clock frequency is 16 MHz
- 011: system clock frequency is 8 MHz *
- 100: system clock frequency is 4 MHz *
- 101: system clock frequency is 2 MHz *
- 110: system clock frequency is 1 MHz *
- 111: not used.

*: If RCC_APB2ENR.MRBLEEN bit is set, writing in CLKSYSDIV one of those values is replaced by a 010b = 16 MHz writing at hardware level.

Warning:
- **if the software programs the 64 MHz frequency target while the RCC_CFGR.HSESEL=1, the hardware switches the system clock tree on HSI64MPLL again (and restart HSIPLL64M analog block if RCC_CFGR.STOPHSI=1)**
- ***To switch the system frequency between 64 / 32 / 16 MHz without risk when the MR_BLE is used, prefer the RCC_CSCMDR register to change the system frequency.***
- **the MR_BLE frequency must always be equal or less than the CPU/system clock to have functional radio.**

Bit 4 Reserved, must be kept at reset value.

Bit 3 **HSESEL_STATUS**: Clock source selection status.
- 0: RC64MPLL clock source is selected (default).
- 1: Direct HSE clock source is selected.

Bit 2 **STOPHSI**: RC64MPLL clock source stop request
- 0: RC64MPLL is enabled (default)
- 1: RC64MPLL disable requested.

Note: if the CLKSYSDIV (from RCC_CFGR or RCC_CSCMDR registers) selects the 64 MHz frequency, the hardware automatically restart the RC64MPLL block and switch on the RC64MPLL clock source.

Bit 1 **HSESEL**: Clock source selection request.
- 0: RC64MPLL clock source is requested (default).
  In this case, the fast clock tree is sourced by the RC64MPLL block. The clock can be either the HSI or the PLL64M if the HSI PLL is locked
- 1: Direct HSE clock source is requested.
  In this case, the RC64MPLL block is not used and the maximum available frequency for the system clock tree is 32 MHz.

Bit 0 Reserved, must be kept at reset value.

1. This bit is reset on PORESETn only.

6.6.3 Clocks source software calibration register (RCC_CSSWCR)

This register allows overloading the trimming values loaded automatically by hardware with other values.

Note: The control bits linked to high speed clock source are reset on PADRESETn. The control bits linked to high speed clock source are reset on PORESETn only (identified by the table footnote).

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	HSITRIMSW[5:0]						HSISWTRIMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSEDRV		LSISWBW[3:0]				LSISWTRIMEN
									rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **HSITRIMSW**: High speed clock trimming set by software.
This value is taken into account instead of the trimming value loaded by HW at reset if HSISWTRIMEN bit is set.

Bit 23 **HSISWTRIMEN**: High speed clock software trimming enable.
- 0: HW trimming value readable in RCC_ICSCR.HSITRIM[3:0] bit field is used as trimming value on RC64MPLL block.
- 1: trimming value written in RCC_CSSWCR.HSITRIMSW[3:0] bit field is used as trimming value on RC64MPLL block.

Bits 22:7 Reserved, must be kept at reset value.

Bits 6:5[1] **LSEDRV**: external 32 kHz crystal GM.
- 00: low drive capability
- 01: medium low drive capability
- 10: medium high drive capability
- 11: high drive capability

Bits 4:1[1] **LSISWBW**: Low speed internal RC trimming value set by software.
This value is taken into account instead of the trimming value loaded by HW at reset if LSISWTRIMEN bit is set.

Bit 0 [1] **LSISWTRIMEN**: Low speed internal RC software trimming enable.
- 0: HW trimming value readable in RCC_ICSCR.LSIBW[3:0] bit field is used as trimming value on the LSI.
- 1: trimming value written in RCC_CSSWCR.LSISWBW[3:0] bit field is used as trimming value on LSI.

1. This bit is reset on PORESETn only.

6.6.4 Clock interrupt enable register (RCC_CIER)

This register controls the enable on interrupt sources.

This register is reset on PADRESETn.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LPURSTIE	WDGRSTIE	RTCRSTIE	HSIPLLUNLOCKDETIE	HSIPLLRDYIE	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSIRDYIE
						rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **LPURSTIE**: LPUART reset release interrupt enable.
- 0: LPUART reset release interrupt is disabled.
- 1: LPUART reset release interrupt is enabled.

Bit 8 **WDGRSTIE**: Watchdog reset release interrupt enable.
- 0: Watchdog reset release interrupt is disabled.
- 1: Watchdog reset release interrupt is enabled.

Bit 7 **RTCRSTIE**: RTC reset release interrupt enable.
- 0: RTC reset release interrupt is disabled.
- 1: RTC reset release interrupt is enabled.

Bit 6 **HSIPLLUNLOCKDETIE**: HSI PLL unlock detection interrupt enable.
- 0: HSI PLL unlocked detection interrupt is disabled.
- 1: HSI PLL unlocked detection is enabled.

Bit 5 **HSIPLLRDYIE**: HSI PLL ready interrupt enable.
- 0: HSI PLL locked interrupt is disabled.
- 1: HSI PLL locked interrupt is enabled.

Bit 4 **HSERDYIE**: HSE ready interrupt enable.
- 0: HSE ready interrupt is disabled.
- 1: HSE ready interrupt is enabled.

Bit 3 **HSIRDYIE**: HSI ready interrupt enable.
- 0: HSI ready interrupt is disabled.
- 1: HSI ready interrupt is enabled.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYIE**: LSE ready interrupt enable.
- 0: LSE ready interrupt is disabled.
- 1: LSE ready interrupt is enabled.

Bit 0 **LSIRDYIE**: LSI ready interrupt enable.
- 0: LSI ready interrupt is disabled.
- 1: LSI ready interrupt is enabled.

6.6.5 Clock interrupt flag register (RCC_CIFR)

This register provides the status flag linked to clock source ready state or not. It is also used to clear the flags.

This register is reset on PADRESETn.

Address offset: 0x1C

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LPURSTF	WDGRSTF	RTCRSTF	HSIPLLUNLOCKDETF	HSIPLLRDYF	HSERDYF	HSIRDYF	Res.	LSERDYF	LSIRDYF
						rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **LPURSTF**: LPUART reset release flag.

– 0: no LPUART reset release event occurred.

– 1: LPUART reset release event occurred.

Cleared by writing 1 in this bit.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the LPUART reset by writing in the RCC_APB0RSTR.LPUARTRST, the reset release is effective only 2 slow clock periods or 2 16 MHz clock periods after the APB writing, depending on how it's configured LPUCLKSEL bit in Clock configuration register (RCC_CFGR) on page 114. This interrupt allows informing the software when the reset release is really done.

Note: this flag is also set after any PORESETn.

Bit 8 **WDGRSTF**: Watchdog reset release flag.

– 0: no Watchdog reset release event occurred.

– 1: Watchdog reset release event occurred.

Cleared by writing 1 in this bit.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the Watchdog reset by writing in the RCC_APB0RSTR.WDGRST, the reset release is effective only 2 slow clock periods after the APB writing. This interrupt allows informing the software when the reset release is really done.

Note: this flag is also set after any PORESETn.

Bit 7 **RTCRSTF**: RTC reset release flag.

– 0: no RTC reset release event occurred.

– 1: RTC reset release event occurred.

Cleared by writing 1 in this bit.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the RTC reset by writing in the RCC_APB0RSTR.RTCRST, the reset release is effective only 2 slow clock periods after the APB writing. This interrupt allows informing the software when the reset release is really done.

Note: this flag is also set after any PORESETn.

Bit 6 **HSIPLLUNLOCKDETF**: HSI PLL unlock detection flag.
- 0: no HSI PLL unlock event occurred.
- 1: HSI PLL unlock event occurred.

Cleared by writing 1 in this bit.

Bit 5 **HSIPLLRDYF**: HSI PLL ready flag.
- 0: no HSI PLL locked event occurred.
- 1: HSI PLL locked event occurred.

Cleared by writing 1 in this bit.

Bit 4 **HSERDYF**: HSE ready flag.
- 0: no HSE ready event occurred.
- 1: HSE ready event occurred.

Cleared by writing 1 in this bit.

Bit 3 **HSIRDYF**: HSI ready flag.
- 0: no HSI ready event occurred.
- 1: HSI ready event occurred.

Cleared by writing 1 in this bit.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYF**: LSE ready flag.
- 0: no LSE ready event occurred.
- 1: LSE ready event occurred.

Cleared by writing 1 in this bit.

Bit 0 **LSIRDYF**: LSI ready flag.
- 0: no LSI ready event occurred.
- 1: LSI ready event occurred.

Cleared by writing 1 in this bit.

6.6.6 Clock switch command register (RCC_CSCMDR)

This register allows switching the CPU / system clock frequency safely while the MR_BLE is active.

Requesting a frequency clock switch holds the AHB/APB transfers between the MR_BLE and the rest of the system to execute safely the clock switching and release AHB / APB transfers as soon as the new frequency is in place.

A dedicated line of interrupt (instead of the RCC line) is used on the NVIC for the EOFSEQ_IRQ information (see *Table 6: Interrupt vectors*).

Note: Anyway, the user must keep the CPU/system frequency at minimum 16 MHz clock when the radio is used.

This register is reset on PADRESETn.

Address offset: 0x20

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOFSEQ_IRQ	EOFSEQ_IE	STATUS[1:0]		CLKSYSDIV_REQ[2:0]			REQUEST
								rc_w1	rw	r	r	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **EOFSEQ_IRQ**: End of sequence flag.
- 0: No end of sequence event occurred.
- 1: End of sequence event occurred.

Cleared by writing 1 in this bit. It's necessary to clear this bit before setting EOFSEQ_IE.

Bit 6 **EOFSEQ_IE**: End of sequence interrupt enable.
- 0: End of sequence interrupt is disabled.
- 1: End of sequence interrupt is enabled.

Bits 5:4 **STATUS**: Status of the switching sequence.
- 00: IDLE = no switch sequence requested / on-going.
- 01: ONGOING = a system clock frequency switch is on-going
- 10: DONE = a system clock frequency switch is done.
- 11: reserved.

This bit field is cleared when EOFSEQ_IRQ bit is cleared.

Bits 3:1 **CLKSYSDIV_REQ**: System clock requested/targeted frequency.
Same format and same notes/warnings as SYSCLKDIV[2:0] bit field described in *6.6.2: Clock configuration register (RCC_CFGR)*.

Bit 0 **REQUEST**: Request to switch the system clock frequency.
Write 1 in this bit to request a system clock frequency switch (using CLKSYSDIV_REQ[2:0] information).
This bit is cleared by hardware when the clock frequency switch is done.
Note: writing 0 in this bit aborts the frequency switch sequence if it is not yet finished. This action must not be used in the normal life of the application except if the end of sequence does not occurs after a long time (to unblock the situation) but this is not supposed to occur.

6.6.7 AHB0 macrocell reset register (RCC_AHBRSTR)

This register allows resetting individually by software each IP located in the AHB0 mapping.

This register is reset on PADRESETn.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRNGRST	Res.	PKARST
													rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOBRST	GPIOARST	Res.	DMARST
			rw									rw	rw		rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **RNGRST**: RNG reset.
- 0: RNG IP is not under reset.
- 1: RNG IP is under reset.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **PKARST**: PKA reset.
- 0: PKA IP is not under reset.
- 1: PKA IP is under reset.

Note: PKA RAM no more accessible by CPU when this bit is set.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset.
- 0: CRC IP is not under reset.
- 1: CRC IP is under reset.

Bits 11:4 Reserved, must be kept at reset value.

Bit 3 **GPIOBRST**: IO controller for port B reset.
- 0: GPIOB IP is not under reset.
- 1: GPIOB IP is under reset.

Bit 2 **GPIOARST**: IO controller for port A reset.
- 0: GPIOA IP is not under reset.
- 1: GPIOA IP is under reset.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMARST**: DMA and DMAMUX reset.
- 0: DMA and DMAMUX IPs are not under reset.
- 1: DMA and DMAMUX IPs are under reset.

6.6.8 APB0 macrocell reset register (RCC_APB0RSTR)

This register allows resetting individually by software each IP located in the APB0 mapping.

This register is reset on PADRESETn.

Note: Each bit is set and reset by the software.

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WDGRST	Res.	RTCRST	Res.	Res.	Res.	SYSCFGRST	Res.	Res.	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM2RST
	rw		rw				rw						rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **WDGRST**: Watchdog reset.
- 0: Watchdog IP is not under reset.
- 1: Watchdog IP is under reset.

Note: due to asynchronism slow clock/fast clock management, when the software requests to release the RTC reset by writing 0 in the RCC_APB0RSTR.RTCRST, the reset release is effective only 2 slow clock periods after the APB writing. An interrupt/status flag is available to inform the software when the reset release is really done (see RCC_CIFR/RCC_CIFR registers).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RTCRST**: RTC reset.
- 0: RTC IP is not under reset.
- 1: RTC IP is under reset.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the RTC reset by writing in the RCC_APB0RSTR.RTCRST, the reset release is effective only 2 slow clock periods after the APB writing. An interrupt/status flag is available to inform the software when the reset release is really done (see RCC_CIFR/RCC_CIFR registers).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SYSCFGRST**: system controller reset.
- 0: system controller IP is not under reset.
- 1: system controller IP is under reset.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **TIM17RST**: TIM17 reset.
- 0: TIM17 IP is not under reset.
- 1: TIM17 IP is under reset.

Bit 1 **TIM16RST**: TIM16 reset.
- 0: TIM16 IP is not under reset.
- 1: TIM16 IP is under reset.

Bit 0 **TIM2RST**: TIM2 reset.
- 0: TIM2 IP is not under reset.
- 1: TIM2 IP is under reset.

6.6.9 APB1 macrocell reset register (RCC_APB1RSTR)

This register allows resetting individually by software each IP located in the APB1 mapping.

This register is reset on PADRESETn.

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1RST	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI3RST	Res.	Res.	Res.	USARTRST	Res.	LPUARTRST	Res.	Res.	Res.	ADCRST	Res.	Res.	Res.	Res.
	rw				rw		rw				rw				

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C1RST**: I2C1 reset.
- 0: I2C1 IP is not under reset.
- 1: I2C1 IP is under reset.

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI3RST**: SPI3 reset.
- 0: SPI3 IP is not under reset.
- 1: SPI3 IP is under reset.

Bit 13:11 Reserved, must be kept at reset value.

Bit 10 **USARTRST**: USART reset.
- 0: USART IP is not under reset.
- 1: USART IP is under reset.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **LPUARTRST**: LPUART reset.
- 0: LPUART IP is not under reset.
- 1: LPUART IP is under reset.

Note: due to asynchronism slow clock/fast clock management, when the software request to release the LPUART reset by writing in the RCC_APB1RSTR.LPUARTRST, the reset release is effective only 2 slow clock periods after the APB writing. An interrupt/status flag is available to inform the software when the reset release is really done (see RCC_CIFR/RCC_CIFR registers).

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ADCRST**: ADC reset.
- 0: ADC IP is not under reset.
- 1: ADC IP is under reset.

Bits 3:0 Reserved, must be kept at reset value.

6.6.10 APB2 macro cells reset register (RCC_APB2RSTR)

This register allows resetting individually by software each IP located in the APB2 mapping (radio).

This register is reset on PADRESETn.

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRBLERST
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MRBLERST**: MR_BLE (Bluetooth radio) reset.
- 0: MR_BLE IP is not under reset.
- 1: MR_BLE IP is under reset.

6.6.11 AHB0 macrocell clock enable register (RCC_AHBENR)

This register allows gating individually by software the clock of each IP located in the AHB0 mapping.

Note: Each IP clock gating is controlled by only 1 bit which gates both AHB clock and kernel clock when the IP uses one.

This register is reset on PADRESETn.

Address offset: 0x50

Reset value: 0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRNGEN	Res.	PKAEN
													rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOBEN	GPIOAEN	Res.	DMAEN
			rw									rw	rw		rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **TRNGEN**: TRNG clock enable.
- 0: TRNG IP is clock gated.
- 1: TRNG IP is clocked.

Bit 16 **PKAEN**: PKA enable.
- 0: PKA IP is clock gated.
- 1: PKA IP is clocked.

*Note: PKA RAM is no more accessible by CPU when this bit i*s set.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC enable.
- 0: CRC IP is clock gated.
- 1: CRC IP is clocked.

Bits 11:4 Reserved, must be kept at reset value.

Bit 3 **GPIOBEN**: IO controller for port Benable.
- 0: GPIOB IP is clock gated.
- 1: GPIOB IP is clocked (default).

Bit 2 **GPIOAEN**: IO controller for port A enable.
- 0: GPIOA IP is clock gated.
- 1: GPIOA IP is clocked (default).

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMAEN**: DMA and DMAMUX enable.
- 0: DMA and DMAMUX IPs are clock gated.
- 1: DMA and DMAMUX IPs are clocked.

6.6.12 APB0 macrocell clock enable register (RCC_APB0ENR)

This register allows gating individually by software the clock of each IP located in the APB0 mapping.

Note: Each IP clock gating is controlled by only 1 bit which gates both APB clock and kernel clock when the IP uses one.

This register is reset on PADRESETn (except one bit identified with a table footnote).

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WDGEN	Res.	RTCEN	Res.	Res.	Res.	SYSCFGEN	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM2EN
	rw		rw				rw						rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **WDGEN**: Watchdog enable.
- 0: Watchdog IP is clock gated.
- 1: Watchdog IP is clocked.

WARNING: the software has to wait 2 slow clock cycles before to use the IWDG IP after setting this bit due to a double resynchronization on slow clock.

Bit 13 Reserved, must be kept at reset value.

Bit 12(1) **RTCEN**: RTC enable.
- 0: RTC IP is clock gated.
- 1: RTC IP is clocked.

WARNING: the software has to wait 2 slow clock cycles before to use the RTC IP after setting this bit due to a double resynchronization on slow clock.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SYSCFGEN**: system controller enable.
- 0: system controller IP is clock gated.
- 1: system controller IP is clocked.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **TIM17EN**: TIM17 enable.
- 0: TIM17 IP is clock gated.
- 1: TIM17 IP is clocked.

Bit 1 **TIM16EN**: TIM16 enable.
- 0: TIM16 IP is clock gated.
- 1: TIM16 IP is clocked.

Bit 0 **TIM2EN**: TIM2 enable.
- 0: TIM2 IP is clock gated.
- 1: TIM2 IP is clocked.

1. This bit is reset on PORESETn only.

6.6.13 APB1 macrocell clock enable register (RCC_APB1ENR)

This register allows gating individually by software the clock of each IP located in the APB1 mapping.

Note: Each IP clock gating is controlled by only 1 bit which gates both APB clock and kernel clock when the IP uses one.

This register is reset on PADRESETn.

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1EN	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI3EN	Res.	Res.	Res.	USARTEN	Res.	LPUARTEN	Res.	Res.	ADCANAEN	ADCDIGEN	Res.	Res.	Res.	Res.
	rw				rw		rw			rw	rw				

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN**: I2C1 enable.
- 0: I2C1 IP is clock gated.
- 1: I2C1 IP is clocked.

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI3EN:** SPI3 enable.
- 0: SPI3 IP is clock gated.
- 1: SPI3 IP is clocked.

Bit 13:11 Reserved, must be kept at reset value.

Bit 10 **USARTEN**: USART enable.
- 0: USART IP is clock gated.
- 1: USART IP is clocked.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **LPUARTEN** LPUART enable.
- 0: LPUART IP is clock gated.
- 1: LPUART IP is clocked.

WARNING: the software has to wait 2 slow clock cycles before to use the LPUART IP after setting this bit due to a double resynchronization on slow clock.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADCANAEN** ADC clock enable for the analog part of the ADC block.
- 0: ADC analog IP is clock gated.
- 1: ADC analog IP is clocked.

Bit 4 **ADCDIGEN** ADC clock enable for digital part of the ADC block.
- 0: ADC digital IP is clock gated.
- 1: ADC digital IP is clocked.

Bits 3:0 Reserved, must be kept at reset value.

6.6.14 APB2 macrocell clock enable register (RCC_APB2ENR)

This register allows gating by software the MR_BLE clock located in the APB2 mapping (radio) and programming the frequency to be used by the MR_BLE IP.

Note: Gating the MR_BLE means both MR_BLE fast and slow clock trees (so including WAKEUP block).

This register is reset on PADRESETn.

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLKBLEDIV	Res.	MRBLEEN
													rw		rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CLKBLEDIV**: MR_BLE (Bluetooth® LE radio) clock frequency selection when RCC_APB2ENR.MRBLEEN=1.reserved.

– 0: 32 MHz.

– 1: 16 MHz.

Warning:

– the MR_BLE frequency must always be equal or less than the CPU/system clock to have functional radio.

– when the ratio between system clock frequency and MR_BLE frequency is modified, the AHBUPCONV block must adapt the clock ratio on APB/AHB bus. Only dynamic CPU system clock switching is managed (see Clock switch command register (RCC_CSCMDR) for details)
For this reason it is strongly recommended to use a static MR_BLE clock configuration.

Bits 1 Reserved, must be kept at reset value.

Bit 0 **MRBLEEN**: MR_BLE (Bluetooth® LE radio) enable.

– 0: MR_BLE IP is clock gated.

– 1: MR_BLE IP is clocked.

6.6.15 Clock control and reset status register (RCC_CSR)

This register provides the reset reason flags. It is set automatically by hardware on any new reset event and must be cleared by software.

Figure 12: Reset generation provides a summary of active flags versus reset reason.

This register is reset on PORESETn.

Address offset: 0x94

Reset value: 0x0C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LOCKUPRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	r	r	r	r	r			rc_w1							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LOCKUPRSTF**: CPU lockup reset flag.
Set by the hardware when a CPU lockup reset occurs.
Reset by writing 1 in RMVF bit.

Bit 29 **WDGRSTF**: Watchdog reset flag.
Set by the hardware when a Watchdog reset occurs.
Reset by writing 1 in RMVF bit.

Bit 28 **SFTRSTF**: Software reset flag.
Set by the hardware when a CPU system reset occurs.
Reset by writing 1 in RMVF bit.

Bit 27 **PORRSTF**: Power-On reset flag.
Set by the hardware when a PORESETN or a BOR reset occurs.
Reset by writing 1 in RMVF bit.

Bit 26 **PADRSTF**: NRSTn pad reset flag.
Set by the hardware when a reset from external NRSTn pad occurs but also after any reset.
This means the source of the reset is the NRSTn pad only if all flags are low except this one.
Reset by writing 1 in RMVF bit.

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: Remove Flag reset.
Writing 1 in this bit clears all the reset flags of this register.
This bit is auto-cleared by the hardware.

Bits 22:0 Reserved, must be kept at reset value.

Note: This register is mirrored at RAM location 0x20000010. In order to get actual register value user must read this RAM location. Direct register read always returns 0

6.6.16 RF software high speed external register (RCC_RFSWHSECR)

This register is reset on PADRESETn.

Address offset: 0x98

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SWXOTUNE[5:0]						SWXOTUNEEN	GMC[2:0]			SATRG	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **SWXOTUNE**: RF HSE capacitor bank tuning value set by software.
This value is taken into account instead of the trimming value loaded by HW at reset if SWXOTUNEEN bit is set.

Bit 7 **SWXOTUNEEN**: RF HSE software capacitor bank tuning enable.
- 0: trimming value readable in RCC_RFHSECR.XOTUNE[5:0] bit field is used as trimming value on HSE.
- 1: trimming value written in RCC_RFSWHSECR.SWXOTUNE[5:0] bit field is used as trimming value on HSE.

Bits 6:4 **GMC**: High speed external IO current control.
- 000: max 0.18 mA/V
- 001: max 0.57 mA/V
- 010: max 0.78 mA/V
- 011: max 1.13 mA/V
- 100: max 0.61 mA/V
- 101: max 1.65 mA/V
- 110: max 2.12 mA/V
- 111: max 2.84 mA/V

Note: this value is set only by software.

Bit 3 **SATRG**: Sense Amplifier Threshold.
- 0: the bias current is confronted to a reference current with a ratio of 1/2.
- 1: the bias current is confronted to a reference current with a ratio of 3/4.

Bits 2:0 Reserved, must be kept at reset value.

6.6.17 RF high speed external register (RCC_RFHSECR)

This register is reset on PADRESETn.

Address offset: 0x9C

Reset value: 0x0000 0000 when STM32WB09xE flash is empty, else depends on trimmed values flashed in the sample.

Note: The XOTUNE value depends on the choice of the external XO component. For this reason, the RCC_RFSWHSECR.SWXOTUNE bit field should be programmed and selected before starting the XO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XOTUNE[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **XOTUNE**: RF-HSE capacitor bank tuning.
This value is loaded by HW at reset as soon as the flash controller achieves the reading of the information in flash memory.

6.6.18 RCC register map

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the RCC base address location in the STM32WB09xE.

The green cells indicate the register is in the V12o power domain and the pink cells indicate the register is in the VBAT (also called V33) power domain. This implies that those registers are not reset on Deepstop exit.

Table 16. RCC register map and reset values

Offset	Register	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
0x00	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDY	HSEON	Res.	HSIPLLRDY	HSIPLLON	HSEPLLBUFON	Res.	HSIRDY	LOCKDET_NSTOP			LSEBYP	LSERDY	LSEON	LSIRDY	LSION	Res.	Res.
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	0	0	1	-	1	0	0	0	0	0	0	0	0	-	-
0x04	Reserved																																
0x08	RCC_CFGR	CCOPRE			MCOSEL			LCOSEL		SPI3I2SCLKSEL		Res.	Res.	LCOEN	IOBOOSTCKEN	IOBOOSTEN	CLKSLOWSEL		Res.	LPUCLKSEL	SMPSDIV	Res.	CLKSYSDIV_STATUS		CLKSYSDIV			Res.	HSESEL_STATUS	STOPHSI	HSESEL	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	-	0	0	-	0	1	0	0	1	0	-	0	0	0	-
0x0C	RCC_CSSWCR	Res.	Res.	HSITRIMSW						HSISWTRIMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSEDRV		LSISWBW				LSISWTRIMEN
	Reset value	-	-	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	0	1	-	-	-	0	0	0	0	0	0	0
0x10 - 0x14	Reserved																																
0x18	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPURSTIE	WDGRSTIE	RTCRSTIE	HSIPLLUNLOCKDETIE	HSIPLLRDYIE	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSIRDYIE
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	-	0	0
0x1C	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPURSTF	WDGRSTF	RTCRSTF	HSIPLLUNLOCKDETF	HSIPLLRDYF	HSERDYF	HSIRDYF	Res.	LSERDYF	LSIRDYF
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	1	-	0	0

Table 16. RCC register map and reset values (continued)

Offset	Register	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
0x20	RCC_CSCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOFSEQ_IRQ	EOFSEQ_IE	STATUS[1:0]		CLKSYSDIV_REQ[2:0]		REQUEST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	0	0	-	0	0
0x24 - 0x2C	Reserved																																
0x30	RCC_AHBRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRNGRST	Res.	PKARST	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOBRS	GPIOARS	Res.	DMARST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	0	-	-	-	0	-	-	-	-	-	-	-	-	0	0	0	0
0x34	RCC_APB0RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGRST	Res.	RTCRST	Res.	Res.	Res.	Res.	SYSCFGRST	Res.	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM2RST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	0	-	-	-	0	0	-	-	-	-	0	0	0
0x38	RCC_APB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1RST	Res.	Res.	Res.	Res.	Res.	Res.	SPI3RST	Res.	Res.	Res.	USARTRST	Res.	LPUARTRST	Res.	Res.	Res.	ADCRST	Res.	Res.	Res.	Res.
	Reset value	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	0	-	-	-	0	-	0	-	-	-	0	-	-	-	-
0x3C	Reserved																																
0x40	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRBLERS
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0x44 - 0x4C	Reserved																																
0x50	RCC_AHBENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRNGEN	Res.	PKAEN	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOBEN	GPIOAEN	Res.	DMAEN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	0	-	-	-	0	-	-	-	-	-	-	-	-	1	1	0	0
0x54	RCC_APB0ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGEN	Res.	RTCEN	Res.	Res.	Res.	Res.	SYSCFGEN	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM2EN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	0	-	-	-	-	0	-	-	-	-	0	0	0

Table 16. RCC register map and reset values (continued)

Offset	Register	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
0x58	RCC_APB1ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1EN	Res.	Res.	Res.	Res.	Res.	Res.	SPI3EN	Res.	Res.	Res.	USARTEN	Res.	LPUARTEN	Res.	Res.	ADCANAEN	ADCDIGEN	Res.	Res.	Res.	Res.
	Reset value	-	-	-	-	-	-	-	-		-	0	-	-	-	-	-	-	0	-		-	0	-	0	-	-	0	0	-	-	-	
0x5C	Reserved																																
0x60	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLKBLEDIV	Res.	MRBLEEN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	0
0x64 - 0x90	Reserved																																
0x94	RCC_CSR	Res.	LOCKUPRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	-	0	0	0	1	1	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x98	RCC_RFSWHSECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWXOTUNE						SWXOTUNEEN	GMC			SATRG	Res.	Res.	Res.
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	1	1	0	-	-	-
0x9C	RCC_RFHSECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XOTUNE					
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

6.7 Programmer's model

6.7.1 Switch the system on the PLL64M clock tree

To switch the system from the HSI clock source to the PLL64M clock source, the user has to:

1. Enable the HSE (32 MHz external crystal)
2. Wait for the HSE ready flag information (through interrupt or by polling)
3. Request to enable the PLL
4. Wait for the PLL ready flag information (through interrupt or by polling). From this point, the clock source for the whole fast clock tree is the accurate PLL64M source.

Note: A status flag and an associated interrupt are available to inform the software in case of HSIPLL64M unlock event. See RCC_CIER and RCC_CIFR registers.

6.7.2 Use the direct HSE instead of the RC64MPLL block

If the application does not target to use the 64 MHz system clock frequency, the system can be configured to use directly the 32 MHz provided by the external XO (HSE).

This configuration choice is supposed to be static and to be used when 64 MHz is never used.

In this case, the software as to:

1. Ensure the RCC_CR.CLKSYSDIV bit field is programmed with a system frequency les than 64 MHz
2. Enable the external XO (by setting the RCC_CR.HSEON if not yet done)
3. Wait for the HSE ready flag information (through interrupt or by polling)
4. Set the RCC_CFGR.HSESEL bit to switch the fast clock tree on HSE path. If both clocks (HSI and HSE) are present, the switch should take around 4 clock cycles.
5. To safe power, the software can stop the RC64MPLL analog block by setting the RCC_CFGR.STOPHSI bit.

Note: A hardware mechanism is in place to restart the RC64MPLL and switch back the clock tree on it if the HSERDY is low or if the CLKSYSDIV bit field has been programmed to request 64 MHz.

The HSE configuration is not lost on a Deepstop sequence. So at wakeup, the system restarts the HSE (thanks to HEON bit) and clock tree switches back on HSE as soon as the HSERDY flag is set. In the meantime, the clock tree runs on the RC64MPLL block. If the STOPHSI was high before the Deepstop, the RC64MPLL analog is switched off as soon as the clock tree is back on HSE path.

Caution: The HSE configuration is not lost on a Deepstop sequence. So at wakeup, the system restarts the HSE (thanks to HEON bit). However, the SW has to switch the system clock back to HSI or PLL64M path to be able to enter in Deepstop/Shutdown so at wakeup:

- the clock tree runs on RC64MPLL block (HSI or PLL64M),
- the SW has to reprogram the HSE mode. However, it is important to avoid the switch of clock tree between HSI and HSE while the MR_BLE already triggered a wakeup event and started its sequence. The SW has to anticipate the CPU wakeup to ensure the final clock source is restored before the radio starts any activity.

6.7.3 Changing the system clock frequency while the MR_BLE is enabled

As long as the MR_BLE is enabled (by setting the RCC_APB2ENR.MRBLEEN), the application software has no guarantee the radio is running or about to start a sequence that makes the MR_BLE IP performing AHB access to the RAM. Changing the system clock and by this action changing the ratio between MR_BLE clock domain and system clock domain could create crash if not managed carefully.

For this reason, a hardware mechanism has been put in place and must be used to change the system frequency when MR_BLE is ON.

The sequence to execute to change the system clock is the following:

1. Ensure the targeted frequency is greater than or equal to the MR_BLE frequency (visible in RCC_APB2ENR.CLKBLEDIV
2. Program the wanted frequency in the RCC_CSCMDR.CLKSYDIV bit field and set the RCC_CSCMDR.REQUEST bit.
3. Wait for the RCC_CSCMDR.EOFSEQ_IRQ flag information (through interrupt or by polling)
4. When the flag (and the interrupt if enabled) is set, the system is running on the new frequency.
5. The RCC_CFGR.CLKSYSDIV[1:0] bit field has been updated by hardware to the new frequency value.

Note: If the software requested a frequency below 16 MHz, the actual final frequency is 16 MHz (associated value is readable in the RCC_CFGR.CLKSYSDIV[1:0].

If the software requests a frequency at 16 MHz while the MR_BLE is clocked at 32 MHz, the requested frequency is used, but the radio scenario no longer functional.

7 General-purpose I/Os (GPIO)

7.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and a 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

7.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every clock cycle
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

7.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the *Section 4: I/O operating modes*, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

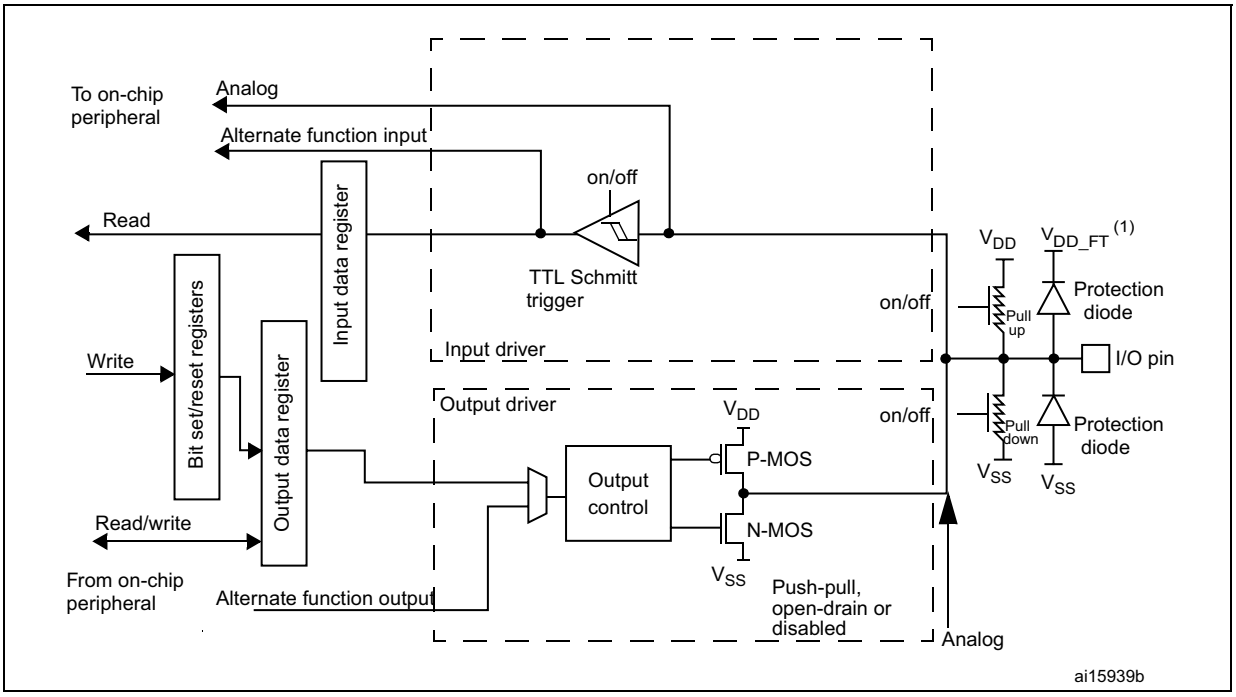
- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Note: Open-drain and analog features are not available on all the I/Os of the STM32WB09xE. *Refer to Table 7: Alternate modes 0, 1 and 2 and Table 8: Alternate modes 3, 4, and 6 footnotes.*

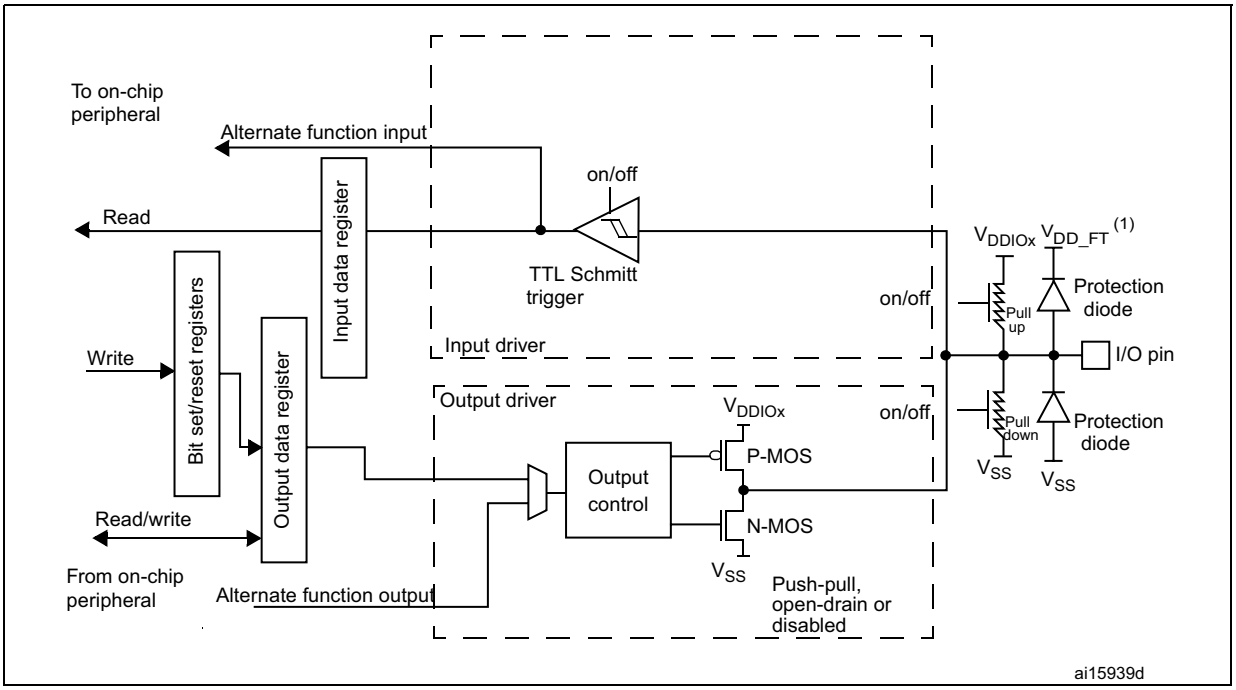
Figure 15 and *Figure 16* show the basic structures of a mixed analog/digital 5 V tolerant I/O port bit and a digital only 5 V tolerant, respectively. *Table 17* gives the possible port bit configurations.

Figure 15. Basic structure of a mixed analog/digital five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Figure 16. Basic structure of a digital only five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 17. Port bit configuration table(1)

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]	PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]	0	0	AF	PP
	0		0	1	AF	PP + PU
	0		1	0	AF	PP + PD
	0		1	1	Reserved	
	1		0	0	AF	OD
	1		0	1	AF	OD + PU
	1		1	0	AF	OD + PD
	1		1	1	Reserved	

Table 17. Port bit configuration table(1) (continued)

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

Note: Open-drain and analog features are not available on all the I/Os of the STM32WB09xE. *Refer to Table 7: Alternate modes 0, 1 and 2 and Table 8: Alternate modes 3, 4, and 6 footnotes.*

7.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in GPIO input pull-up mode except the SWD debug pins.

The debug pins are in AF0 pull-up/pull-down after reset:

- PA2: DEBUG_SWDIO in pull-up
- PA3: DEBUG_SWDCLK in pull-down

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

7.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to six alternate function inputs (AF0, AF1, AF2, AF3, AF4, AF6) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the *Table 4: I/O operating modes*.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- **Debug function:** after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **System function:** RCC_MCO and RCC_LCO pins have to be configured in alternate function mode.
- **GPIO:** configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Alternate function:**
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
  - **Cortex-M0+ alternate function (EVENTOUT)**: The Cortex®-M0+ output EVENTOUT signal can be output as alternate function on several I/Os. An event can be signaled through the configured pin after executing an SEV instruction.
- **Additional functions:**
  - For the ADC, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADCregisters.

Note: 1 When configuring IOs in analog mode, the user must disable the pull-up/pull-down through the PWRC registers, if PWRC_CR1.APC is set.

2 If PWRC_CR1.APC is set (default configuration), the pull-up/-down of the I/Os is controlled by the PWRC_PUCRx and PWRC_PDCRx registers of the PWRC block. Otherwise it is controlled by the GPIOx_PUPDR register of the GPIO block.

- For the additional functions like Wakeup I/Os, LSE oscillator, RCC_LCO configure the required function in the related PWRC, RCC, and RTC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the *Table 10: I/O Additional function mapping* for the detailed mapping of the alternate function I/O pins.

7.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-

pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

7.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

See *Section 7.4.9: GPIOA port input data register (GPIOA_IDR)* and *Section 7.4.11: GPIOA port output data register (GPIOA_ODR)* for the register descriptions.

7.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BR(i) **resets** the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

7.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence (refer to *Section 7.4.14: GPIOB port bit set/reset register (GPIOB_BSRR)*) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in *Section 7.4.14: GPIOB port bit set/reset register (GPIOB_BSRR)*.

7.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, you can connect an alternate function to some other pin as required by your application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the *Table 4: I/O operating modes*.

7.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the IO port must be configured in input mode. The interruption configuration (level/edge, polarity, mask)has to be done in the system controller (SYSCFG). See *Section 8: System controller (SYSCFG)*.

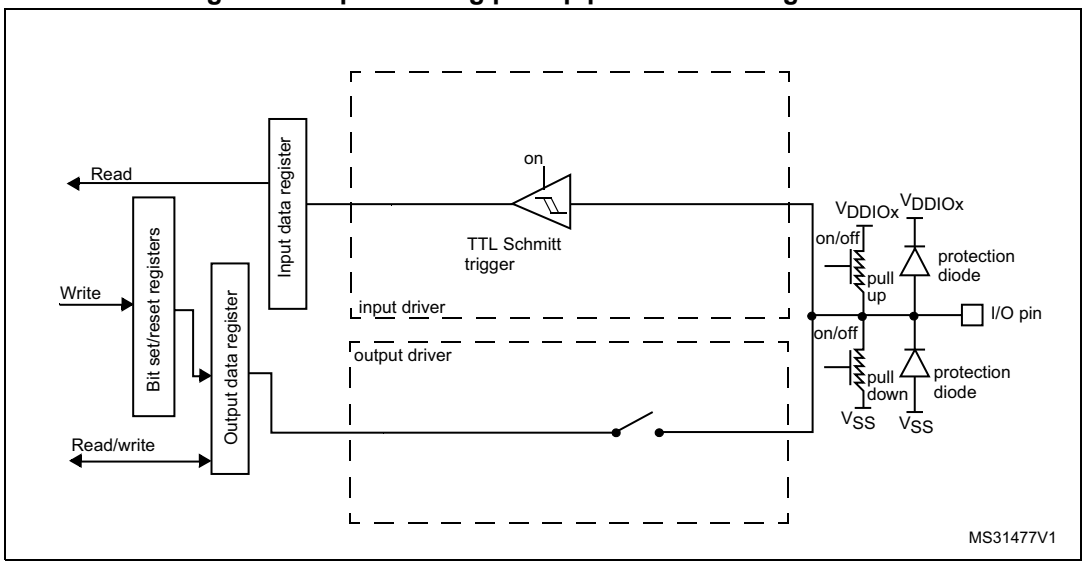
7.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

Figure 17 shows the input configuration of the I/O port bit.

Figure 17. Input floating/pull up/pull down configurations



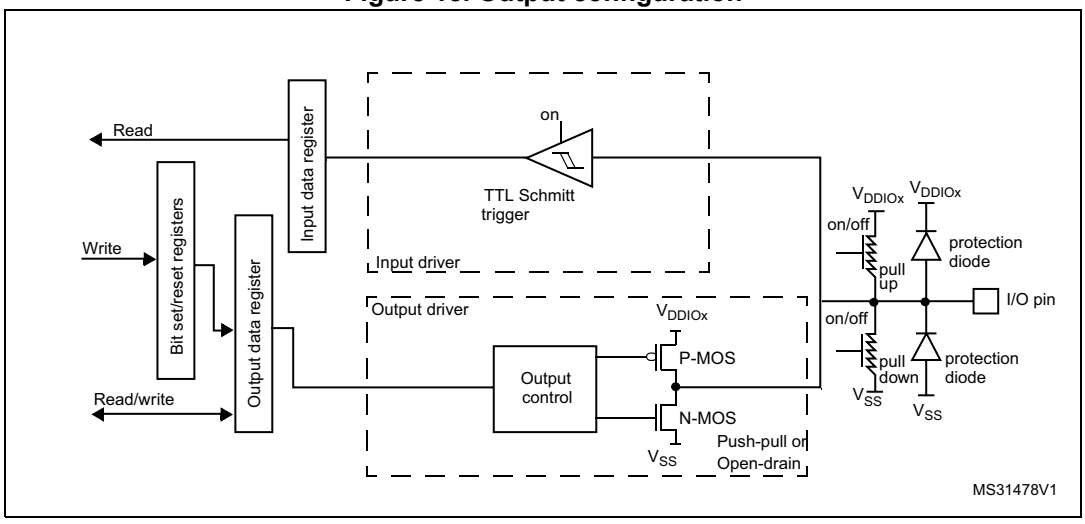
7.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 18 shows the output configuration of the I/O port bit.

Figure 18. Output configuration



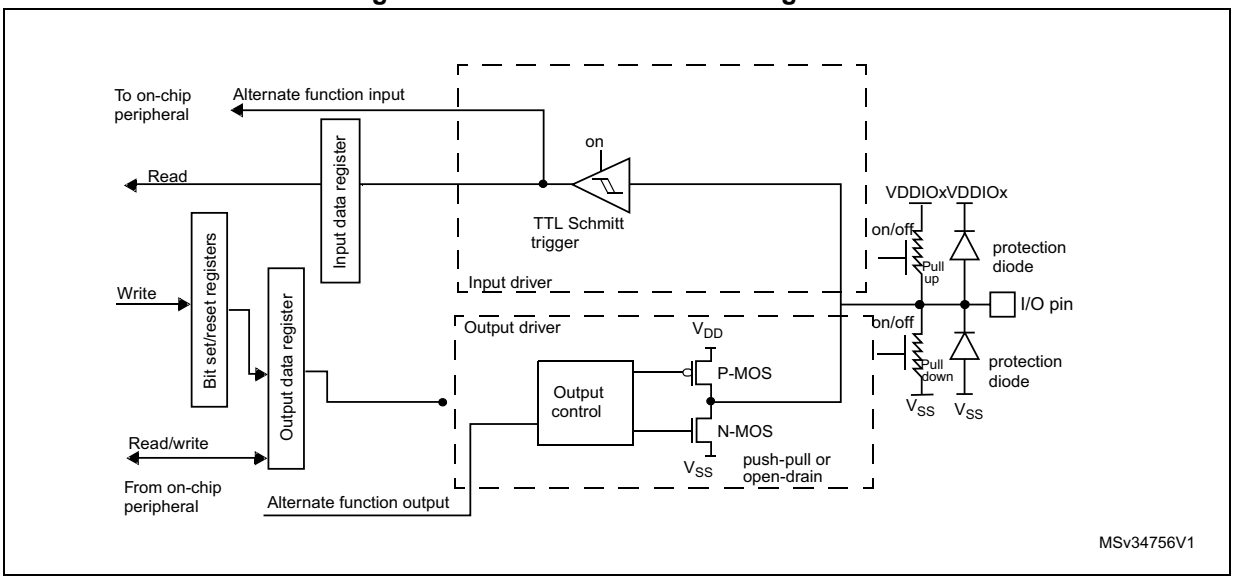
7.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 19 shows the Alternate function configuration of the I/O port bit.

Figure 19. Alternate function configuration



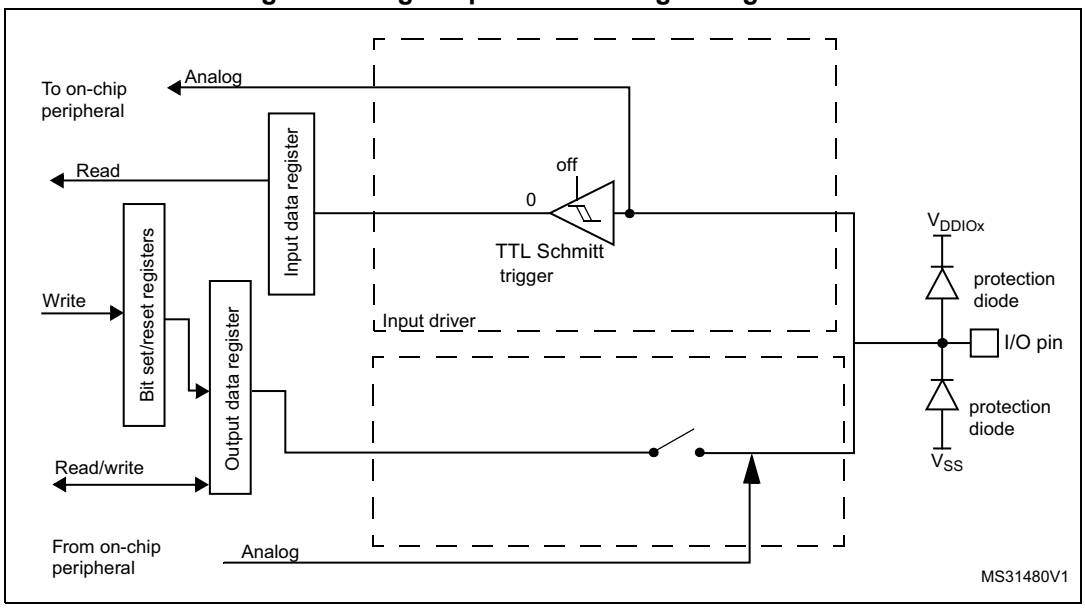
7.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The pull-up and pull-down resistors have to be disabled by the software else the associated analog feature does not work as expected.
- Read access to the input data register gets the value “0”

Figure 20 shows the high-impedance, analog-input configuration of the I/O port bit.

Figure 20. High impedance-analog configuration



7.3.13 Using the LSE oscillator pins as GPIOs

When the LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the oscillator is configured in a user external clock mode, only the OSC32_IN pin is reserved for clock input, automatically if LSEBYP bit is set, and the OSC32_OUT pin can still be used as normal GPIO.

Note: The high speed oscillator (HSE) OSC_IN and OSC_OUT pins are dedicated oscillator pins and can not be used as GPIO.

7.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to *Table 18*.

The peripheral registers can be written in word, half word or byte mode.

7.4.1 GPIOA port mode register (GPIOA_MODER)

Address offset:0x00

Reset values:

- 0x0000 00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODEy[1:0]:** Port A configuration bits (y = 0..15)
These bits are written by software to configure the I/O mode.
00: Input mode
01: output mode
10: Alternate function mode
11: Analog mode
Note: when configuring a pad in Analog mode, the user must take care to disable the associated pull-up/down to avoid pollution on the analog signal.

7.4.2 GPIOB port mode register (GPIOB_MODER)

Address offset:0x00

Reset values:

- 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODEy[1:0]:** Port B configuration bits (y = 0..15)
These bits are written by software to configure the I/O mode.
00: Input mode
01: output mode
10: Alternate function mode
11: Analog mode
Note: when configuring a pad in Analog mode, the user must take care to disable the associated pull-up/down to avoid pollution on the analog signal.

7.4.3 GPIOA port output type register (GPIOA_OTYPER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OT11	OT10	OT9	OT8	Res.	Res.	Res.	Res.	OT3	OT2	OT1	OT0
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port A configuration bits (y = 0..15)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

7.4.4 GPIOB port output type register (GPIOB_OTYPER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	Res.	Res.	Res.	Res.	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port B configuration bits (y = 0..15)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

7.4.5 GPIOA port output speed register (GPIOA_OSPEEDR)

Address offset: 0x08

Reset value:

- 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **OSPEEDy[1:0]**: Port A configuration bits (y = 0..15)
These bits are written by software to configure the I/O output speed.
Note:

7.4.6 GPIOB port output speed register (GPIOB_OSPEEDR)

Address offset: 0x08

Reset value:

- 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **OSPEEDy[1:0]**: Port B configuration bits (y = 0..15)
These bits are written by software to configure the I/O output speed.
Note:

7.4.7 GPIOA port pull-up/pull-down register (GPIOA_PUPDR)

Address offset: 0x0C

Reset values:

- 0x0055 0095

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **PUPDy[1:0]:** Port A configuration bits (y = 0..15)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

7.4.8 GPIOB port pull-up/pull-down register (GPIOB_PUPDR)

Address offset: 0x0C

Reset values:

- 0x5500 5555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **PUPDy[1:0]:** Port B configuration bits (y = 0..15)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

7.4.9 GPIOA port input data register (GPIOA_IDR)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ID11	ID10	ID9	ID8	Res.	Res.	Res.	Res.	ID3	ID2	ID1	ID0
				r	r	r	r					r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port A input data bit (y = 0..15)
These bits are read-only. They contain the input value of the corresponding I/O port.

7.4.10 GPIOB port input data register (GPIOB_IDR)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	Res.	Res.	Res.	Res.	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r					r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port B input data bit (y = 0..15)
These bits are read-only. They contain the input value of the corresponding I/O port.

7.4.11 GPIOA port output data register (GPIOA_ODR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OD11	OD10	OD9	OD8	Res.	Res.	Res.	Res.	OD3	OD2	OD1	OD0
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port A output data bit (y = 0..15)
These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A, B)

7.4.12 GPIOB port output data register (GPIOB_ODR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	Res.	Res.	Res.	Res.	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port B output data bit (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A, B)

7.4.13 GPIOA port bit set/reset register (GPIOA_BSRR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	BR11	BR10	BR9	BR8	Res.	Res.	Res.	Res.	BR3	BR2	BR1	BR0
				w	w	w	w					w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BS11	BS10	BS9	BS8	Res.	Res.	Res.	Res.	BS3	BS2	BS1	BS0
				w	w	w	w					w	w	w	w

Bits 31:16 **BRy:** Port A reset bit y (y = 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port A set bit y (y= 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Sets the corresponding ODx bit

7.4.14 GPIOB port bit set/reset register (GPIOB_BSRR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	Res.	Res.	Res.	Res.	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w					w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	Res.	Res.	Res.	Res.	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w					w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port B reset bit y (y = 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port B set bit y (y= 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Sets the corresponding ODx bit

7.4.15 GPIOA port configuration lock register (GPIOA_LCKR)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
Res.	Res.	Res.	Res.	11	10	9	8	Res.	Res.	Res.	Res.	3	2	1	0
				LCK11	LCK10	LCK9	LCK8					LCK3	LCK2	LCK1	LCK0
Res.	Res.	Res.	Res.	rw	rw	rw	rw	Res.	Res.	Res.	Res.	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = '1' + LCKR[15:0]

WR LCKR[16] = '0' + LCKR[15:0]

WR LCKR[16] = '1' + LCKR[15:0]

RD LCKR

RD LCKR[16] = '1' (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns '1' until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port A lock bit y (y= 0..15)

These bits are read/write but can only be written when the LCKK bit is '0, using the specific sequence described in LCKK bit description.

0: Port configuration not locked

1: Port configuration locked

7.4.16 GPIOB port configuration lock register (GPIOB_LCKR)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	Res.	Res.	Res.	Res.	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = '1' + LCKR[15:0]

WR LCKR[16] = '0' + LCKR[15:0]

WR LCKR[16] = '1' + LCKR[15:0]

RD LCKR

RD LCKR[16] = '1' (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns '1' until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port B lock bit y (y= 0..15)

These bits are read/write but can only be written when the LCKK bit is '0, using the specific sequence described in LCKK bit description.

0: Port configuration not locked

1: Port configuration locked

7.4.17 GPIOA alternate function low register (GPIOA_AFRL)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **y[3:0]:** Alternate function selection for port A pin y (y = 0..7)

These bits are written by software to configure alternate function I/Os

AFSELy selection:

0000: AF0

0001: AF1

0010: AF2

0011: AF3

0100: AF4

0110: AF6

Others: Reserved

7.4.18 GPIOB alternate function low register (GPIOB_AFRL)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **y[3:0]:** Alternate function selection for port B pin y (y = 0..7)
These bits are written by software to configure alternate function I/Os

AFSELy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0110: AF6
Others: Reserved

7.4.19 GPIOA alternate function high register (GPIOA_AFRH)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **y[3:0]:** Alternate function selection for port A pin y (y = 8..15)
These bits are written by software to configure alternate function I/Os

AFSELy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0110: AF6
Others: Reserved

7.4.20 GPIOB alternate function high register (GPIOB_AFRH)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 **y[3:0]:** Alternate function selection for port B pin y (y = 8..15)
These bits are written by software to configure alternate function I/Os

AFSELy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0110: AF6
Others: Reserved

7.4.21 GPIOA port bit reset register (GPIOA_BRR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BR11	BR10	BR9	BR8	Res.	Res.	Res.	Res.	BR3	BR2	BR1	BR0
				w	w	w	w					w	w	w	w

Bits 31:16 **BRy:** Port A reset bit y (y = 0..15)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODx bit
1: Resets the corresponding ODx bit

7.4.22 GPIOB port bit set/reset register (GPIOB_BSRR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	Res.	Res.	Res.	Res.	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w					w	w	w	w	w	w	w	w

Bits 15:0 **BRy:** Port B reset bit y (y = 0..15)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

7.4.23 GPIO register map

The following table gives the GPIO register map and reset values.

Table 18. GPIO register map and reset values

Offset	Register	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
0x00	**GPIOA_MODER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	1	0	1	0	0	0	0	0
0x00	**GPIOB_MODER**	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**GPIOA_OTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT11	OT10	OT9	OT8	Res.	Res.	Res.	Res.	OT3	OT2	OT1	OT0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0
0x04	**GPIOB_OTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	Res.	Res.	Res.	Res.	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0
0x08	**GPIOA_OSPEEDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	1	1	0	0	0	0
0x08	**GPIOB_OSPEEDR**	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	-	-	-	-	-	-	-	-	0	1	0	1	0	1	0	1	-	-	-	-	-	-	-	-	1	0	0	1	0	1	0	1
0x0C	**GPIOB_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	0	1	0	1	0	1	-	-	-	-	-	-	-	-	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1
0x10	**GPIOA_IDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID11	ID10	ID9	ID8	Res.	Res.	Res.	Res.	ID3	ID2	ID1	ID0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	-	-	-	-	x	x	x	x
0x10	**GPIOB_IDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	Res.	Res.	Res.	Res.	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	-	-	-	-	x	x	x	x	x	x	x	x

Table 18. GPIO register map and reset values (continued)

Offset	Register	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
0x14	**GPIOA_ODR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD11	OD10	OD9	OD8	Res.	Res.	Res.	Res.	OD3	OD2	OD1	OD0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0
0x14	**GPIOB_ODR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	Res.	Res.	Res.	Res.	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0
0x18	**GPIOA_BSRR**	Res.	Res.	Res.	Res.	BR11	BR10	BR9	BR8	Res.	Res.	Res.	Res.	BR3	BR2	BR1	BR0	Res.	Res.	Res.	Res.	BS11	BS10	BS9	BS8	Res.	Res.	Res.	Res.	BS3	BS2	BS1	BS0
	Reset value	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0
0x18	**GPIOB_BSRR**	BR15	BR14	BR13	BR12	Res.	Res.	Res.	Res.	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	Res.	Res.	Res.	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	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
0x1C	**GPIOA_LCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	Res.	Res.	Res.	Res.	LCK11	LCK10	LCK9	LCK8	Res.	Res.	Res.	Res.	LCK3	LCK2	LCK1	LCK0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0
0x1C	**GPIOB_LCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	Res.	Res.	Res.	Res.	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0
0x20	**GPIOA_AFRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOB_AFRL**	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	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
0x24	**GPIOA_AFRH** (where x = A, B)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOB_AFRH** (where x = A, B)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x28	**GPIOA_BRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR11	BR10	BR9	BR8	Res.	Res.	Res.	Res.	BR3	BR2	BR1	BR0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0
0x28	**GPIOB_BRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	Res.	Res.	Res.	Res.	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

8 System controller (SYSCFG)

The System Controller is a set of registers (configuration, control and status) linked to system features of the STM32WB09xE device.

8.1 SYSCFG main features

The System Controller set of registers are mainly linked to:

- Provide the JTAG_ID, Die ID and cut version
- Manage the interrupts linked to GPIO feature
- Manage the interrupts linked to the Power Controller (PWRC)
- Manage the interrupt linked to MR_BLE reception and transmission sequences
- Enable/disable I2C Fast-mode Plus driving capability on some I/Os

Note: This peripheral is in the non-retained power domain so all settings done in the associated registers are lost after a Deepstop.

8.2 System controller registers

8.2.1 Die ID register (DIE_ID)

This register provides the device version and cut information.

Address offset: 0x00

Reset value: 0x0000 0110

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PRODUCT[3:0]				VERSION[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **PRODUCT**: Product version.STM32WB09xE

Bits 7:4 **VERSION**: Cut version.

Bits 3:0 **REVISION**: Cut revision (metal fix)

8.2.2 JTAG ID register (JTAG_ID)

This register provides the JTAG ID of the STM32WB09xE.

Note: The same information is also available through direct SWD access to test registers.

Address offset: 0x04

Reset value: 0x0203 2041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VERSION_NUMBER[3:0]				PART_NUMBER[15:4]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PART_NUMBER[3:0]				MANUF_ID[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **VERSION_NUMBER**: Version.

Bits 27:12 **PART_NUMBER**: part number.

Bits 11:1 **MANUF_ID**: Manufacturer ID.

Bit 0 **RESERVED**

8.2.3 Fast-Mode Plus pin capability control register (I2C_FMP_CTRL)

This register allows activating the Fast-mode Plus driving capability on I2C open-drain pads.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1_PB7_FMP	I2C1_PB6_FMP	I2C1_PA1_FMP	I2C1_PA0_FMP
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **I2C1_PB7_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SDA on PB7 I/O.
- 0: PB7 pin operated in standard mode.
- 1: FM+ mode is enabled on PB7 pin, and speed control is bypassed.

Bit 2 **I2C1_PB6_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SCL on PB6 I/O.
- 0: PB6 pin operated in standard mode.
- 1: FM+ mode is enabled on PB6 pin, and speed control is bypassed.

Bit 1 **I2C1_PA1_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SDA on PA1 I/O.
- 0: PA1 pin operated in standard mode.
- 1: FM+ mode is enabled on PA1 pin, and speed control is bypassed.

Bit 0 **I2C1_PA0_FMP**: I2C1 Fast-Mode Plus driving capability for I2C1_SCL on PA0 I/O.
- 0: PA0 pin operated in standard mode.
- 1: FM+ mode is enabled on PA0 pin, and speed control is bypassed.

8.2.4 I/O Interrupt detection type register (IO_DTR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_DT	PB14_DT	PB13_DT	PB12_DT	Res.	Res.	Res.	Res.	PB7_DT	PB6_DT	PB5_DT	PB4_DT	PB3_DT	PB2_DT	PB1_DT	PB0_DT
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PA11_DT	PA10_DT	PA9_DT	PA8_DT	Res.	Res.	Res.	Res.	PA3_DT	PA2_DT	PA1_DT	PA0_DT
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:16 **PBx_DT** (x=15 to 0): Interrupt Detection Type for port B I/Os.
- 0: edge detection.
- 1: level detection.

Bits 15:0 **PAx_DT** (x=15 to 0): Interrupt Detection Type for Port A I/Os.
- 0: edge detection.
- 1: level detection.

8.2.5 I/O Interrupt Edge register (IO_IBER)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_IBE	PB14_IBE	PB13_IBE	PB12_IBE	Res.	Res.	Res.	Res.	PB7_IBE	PB6_IBE	PB5_IBE	PB4_IBE	PB3_IBE	PB2_IBE	PB1_IBE	PB0_IBE
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PA11_IBE	PA10_IBE	PA9_IBE	PA8_IBE	Res.	Res.	Res.	Res.	PA3_IBE	PA2_IBE	PA1_IBE	PA0_IBE
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:16 **PBx_IBE** (x=15 to 0): Interrupt edge selection for port B I/Os.
- 0: single edge detection.
- 1: both edges detection.

Bits 15:0 **PAx_IBE** (x=15 to 0): Interrupt edge selection for Port A I/Os.
- 0: single edge detection.
- 1: both edges detection.

8.2.6 I/O Interrupt polarity event register (IO_IEVR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_IEV	PB14_IEV	PB13_IEV	PB12_IEV	Res.	Res.	Res.	Res.	PB7_IEV	PB6_IEV	PB5_IEV	PB4_IEV	PB3_IEV	PB2_IEV	PB1_IEV	PB0_IEV
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PA11_IEV	PA10_IEV	PA9_IEV	PA8_IEV	Res.	Res.	Res.	Res.	PA3_IEV	PA2_IEV	PA1_IEV	PA0_IEV
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:16 **PBx_IEV** (x=15 to 0): Interrupt polarity event for port B I/Os.
- 0: falling edge / low level.
- 1: rising edge / high level.

Bits 15:0 **PAx_IEV** (x=15 to 0): Interrupt polarity event for Port A I/Os.
- 0: falling edge / low level.
- 1: rising edge / high level.

8.2.7 I/O Interrupt Enable register (IO_IER)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_IE	PB14_IE	PB13_IE	PB12_IE	Res.	Res.	Res.	Res.	PB7_IE	PB6_IE	PB5_IE	PB4_IE	PB3_IE	PB2_IE	PB1_IE	PB0_IE
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PA11_IE	PA10_IE	PA9_IE	PA8_IE	Res.	Res.	Res.	Res.	PA3_IE	PA2_IE	PA1_IE	PA0_IE
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:16 **PBx_IE** (x=15 to 0): Interrupt enable for port B I/Os.
- 0: interrupt is disabled.
- 1: interrupt is enabled.

Bits 15:0 **PAx_IE** (x=15 to 0): Interrupt enable for Port A I/Os.
- 0: interrupt is disabled.
- 1: interrupt is enabled.

8.2.8 I/O Interrupt Status and Clear register (IO_ISCR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PB15_ISC	PB14_ISC	PB13_ISC	PB12_ISC	Res.	Res.	Res.	Res.	PB7_ISC	PB6_ISC	PB5_ISC	PB4_ISC	PB3_ISC	PB2_ISC	PB1_ISC	PB0_ISC
rc_w1	rc_w1	rc_w1	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PA11_ISC	PA10_ISC	PA9_ISC	PA8_ISC	Res.	Res.	Res.	Res.	PA3_ISC	PA2_ISC	PA1_ISC	PA0_ISC
				rc_w1	rc_w1	rc_w1	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 **PBx_ISC** (x=15 to 0): Interrupt status (before mask) for port B I/Os.
- 0: no pending interrupt.
- 1: event occurred on corresponding I/O / interrupt occurred (if enabled).

Cleared by writing 1 in the bit.

Bits 15:0 **PAx_ISC** (x=15 to 0): Interrupt status (before mask) for port A I/Os.
- 0: no pending interrupt.
- 1: event occurred on corresponding I/O / interrupt occurred (if enabled).
- Cleared by writing 1 in the bit.

8.2.9 Power Controller Interrupt Enable register (PWRC_IER)

This register allows control of the enable or mask on the interrupt sources of the Power Controller (PWRC) block.

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.										WKUP_IE	PVD_IE	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WKUP_IE**: Power Controller Wakeup event interrupt enable.
- 0: Interrupt on wakeup event seen by the PWRC is disabled.
- 1: Interrupt on wakeup event seen by the PWRC is enabled.

Bit 1 **PVD_IE**: Programmable Voltage Detector interrupt enable.
- 0: PVD interrupt is disabled.
- 1: PVD interrupt is enabled.

Bit 0 Reserved, must be kept at reset value.

8.2.10 Power Controller Interrupt Status and Clear register (PWRC_ISCR)

This register allows checking the status and clear the interrupt sources of the Power Controller (PWRC) block.

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUP_ISC	PVD_ISC	Res.
													rc_w1	rc_w1	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WKUP_ISC**: Indicates the Power Controller receives a Wakeup event.
- 0: no pending interrupt.
- 1: Wakeup event on PWRC occurred / interrupt occurred (if enabled).

Cleared by writing 1 in the bit.

This flag is read at 1 if a wakeup event arrives so close to the low power mode entry requests that the PWRC aborts before shutting down the system.

Bit 1 **PVD_ISC**: Programmable Voltage Detector status.
- 0: no pending interrupt.
- 1: voltage went under programmed threshold / interrupt occurred (if enabled).

Cleared by writing 1 in the bit.

See *Section 5.3.2: Programmable voltage detectior (PVD)* for details.

Bit 0 Reserved, must be kept at reset value.

8.2.11 IMR_BLE RX or TX sequence information detection type register (BLERXTX_DTR)

This register allows selecting the type of detection (level or edge) on RADIO_RX_SEQUENCE and RADIO_TX_SEQUENCE coming from the MR_BLE IP.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_DT	TX_DT
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RX_DT**: detection type on RADIO_RX_SEQUENCE signal:
- 0: detection on edge (default).
- 1: detection on level

Bit 0 **TX_DT**: detection type on RADIO_TX_SEQUENCE signal:
- 0: detection on edge (default).
- 1: detection on level

8.2.12 MR_BLE RX or TX sequence information detection type register (BLERXTX_IBER)

This register is used to activate RADIO_RX_SEQUENCE and RADIO_TX_SEQUENCE signal detection on single edge or both edges when edge detection type is active.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_IBE	TX_IBE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RX_IBE**: interrupt edge register on RADIO_RX_SEQUENCE signal:
- 0: detection on single edge (default).
- 1: detection on both edges

Bit 0 **TX_IBE**: interrupt edge register on RADIO_TX_SEQUENCE signal:
- 0: detection on single edge (default).
- 1: detection on both edges

8.2.13 MR_BLE RX or TX sequence information detection event register (BLERXTX_IEVR)

This register defines the polarity of the RADIO_RX_SEQUENCE and RADIO_TX_SEQUENCE signals detection.

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_IEV	TX_IEV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RX_IEV**: interrupt polarity event on RADIO_RX_SEQUENCE signal:
- 0: detection on falling edge / low level (default).
- 1: detection on rising edge / high level

Bit 0 **TX_IEV**: interrupt polarity event on RADIO_TX_SEQUENCE signal:
- 0: detection on falling edge / low level (default).
- 1: detection on rising edge / high level

8.2.14 MR_BLE RX or TX sequence information detection interrupt enable register (BLERXTX_IER)

This register enable/disable interrupt generation on the RADIO_RX_SEQUENCE and RADIO_TX_SEQUENCE signals detection.

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_IE	TX_IE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RX_IE**: interrupt enable on RADIO_RX_SEQUENCE signal:
- 0: RADIO_RX_SEQUENCE interrupt is disabled (default).
- 1: RADIO_RX_SEQUENCE interrupt is enabled

Bit 0 **TX_IE**: interrupt enable on RADIO_TX_SEQUENCE signal:
- 0: RADIO_TX_SEQUENCE interrupt is disabled (default).
- 1: RADIO_TX_SEQUENCE interrupt is enabled

8.2.15 MR_BLE RX or TX sequence information detection status and clear register (BLERXTX_ISCR)

This register allows checking the status and clear the interrupt linked to the RADIO_RX_SEQUENCE and RADIO_TX_SEQUENCE information provided by the MR_BLE IP.

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_IS EDGE	TX_IS EDGE	RX_ISC	TX_ISC
												r	r	rc_wc1	rc_wr1

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RX_ISEDGE**: interrupt edge status on RADIO_RX_SEQUENCE signal:
- 0: falling edge on RADIO_RX_SEQUENCE detected.
- 1: rising edge on RADIO_RX_SEQUENCE detected.

Bit 2 **TX_ISEDGE**: interrupt edge status on RADIO_TX_SEQUENCE signal:
- 0: falling edge on RADIO_TX_SEQUENCE detected.
- 1: rising edge on RADIO_TX_SEQUENCE detected.

Bit 1 **RX_ISC**: interrupt status on RADIO_RX_SEQUENCE signal (can be a rising or a falling edge depending on BLERXTX_IEVR and BLERXTX_IBER):
- 0: no activity on RADIO_RX_SEQUENCE detected.
- 1: activity on RADIO_RX_SEQUENCE occurred.

Bit 0 **TX_ISC**: interrupt status on RADIO_TX_SEQUENCE signal (can be a rising or a falling edge depending on BLERXTX_IEVR and BLERXTX_IBER):
- 0: no activity on RADIO_TX_SEQUENCE detected.
- 1: activity on RADIO_TX_SEQUENCE occurred.

8.2.16 System controller register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the System controller base address location in the STM32WB09xE.

Table 19. SYSCFG register map and reset values

Offset	Register	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
0x00	DIE_ID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRODUCT				VERSION				REVISION			
	Reset value																					0	0	0	1	0	0	0	1	0	0	0	0
0x04	JTAG_ID	VERSION_NUMBER				PART_NUMBER																MANUF_ID											Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1
0x08	I2C_FMP_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1_PB7_FMP	I2C1_PB6_FMP	I2C1_PA1_FMP	I2C1_PA0_FMP
	Reset value																													0	0	0	0
0x0C	IO_DTR	PB15_DT	PB14_DT	PB13_DT	PB12_DT	Res.	Res.	Res.	Res.	PB7_DT	PB6_DT	PB5_DT	PB4_DT	PB3_DT	PB2_DT	PB1_DT	PB0_DT	Res.	Res.	Res.	Res.	PA11_DT	PA10_DT	PA9_DT	PA8_DT	Res.	Res.	Res.	Res.	PA3_DT	PA2_DT	PA1_DT	PA0_DT
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0					0	0	0	0					0	0	0	0
0x10	IO_IBER	PB15_IBE	PB14_IBE	PB13_IBE	PB12_IBE	Res.	Res.	Res.	Res.	PB7_IBE	PB6_IBE	PB5_IBE	PB4_IBE	PB3_IBE	PB2_IBE	PB1_IBE	PB0_IBE	Res.	Res.	Res.	Res.	PA11_IBE	PA10_IBE	PA9_IBE	PA8_IBE	Res.	Res.	Res.	Res.	PA3_IBE	PA2_IBE	PA1_IBE	PA0_IBE
	Reset value	0	0	0	0					0		0	0	0	0	0	0					0	0	0	0					0	0	0	0
0x14	IO_IEVR	PB15_IEV	PB14_IEV	PB13_IEV	PB12_IEV	Res.	Res.	Res.	Res.	PB7_IEV	PB6_IEV	PB5_IEV	PB4_IEV	PB3_IEV	PB2_IEV	PB1_IEV	PB0_IEV	Res.	Res.	Res.	Res.	PA11_IEV	PA10_IEV	PA9_IEV	PA8_IEV	Res.	Res.	Res.	Res.	PA3_IEV	PA2_IEV	PA1_IEV	PA0_IEV
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0					0	0	0	0					0	0	0	0
0x18	IO_IER	PB15_IE	PB14_IE	PB13_IE	PB12_IE	Res.	Res.	Res.	Res.	PB7_IE	PB6_IE	PB5_IE	PB4_IE	PB3_IE	PB2_IE	PB1_IE	PB0_IE	Res.	Res.	Res.	Res.	PA11_IE	PA10_IE	PA9_IE	PA8_IE	Res.	Res.	Res.	Res.	PA3_IE	PA2_IE	PA1_IE	PA0_IE
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0					0	0	0	0					0	0	0	0
0x1C	IO_ISCR	PB15_ISC	PB14_ISC	PB13_ISC	PB12_ISC	Res.	Res.	Res.	Res.	PB7_ISC	PB6_ISC	PB5_ISC	PB4_ISC	PB3_ISC	PB2_ISC	PB1_ISC	PB0_ISC	Res.	Res.	Res.	Res.	PA11_ISC	PA10_ISC	PA9_ISC	PA8_ISC	Res.	Res.	Res.	Res.	PA3_ISC	PA2_ISC	PA1_ISC	PA0_ISC
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0					0	0	0	0					0	0	0	0
0x20	PWRC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUP_IE	PVD_IE	Res.
	Reset value																														0	0	

Table 19. SYSCFG register map and reset values (continued)

Offset	Register	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
0x24	PWRC_ISCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUP_ISC	PVD_ISC	Res.
	Reset value																														0	0	
0x28	Reserved																																
0x2C	BLERXTX_DTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_DT	TX_DT
	Reset value																															0	0
0x30	BLERXTX_IBER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_IBE	TX_IBE
	Reset value																															0	0
0x34	BLERXTX_IEVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_IEV	TX_IBV
	Reset value																															0	0
0x38	BLERXTX_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_IE	TX_IB
	Reset value																															0	0
0x3C	BLERXTX_ISCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_ISEDGE	TX_ISEDGE	RX_ISC	TX_ISC
	Reset value																													0	0	0	0

9 Embedded flash memory (FLASH)

The flash controller implements the erase and program flash memory operation. The flash controller also implements the read and write protection.

9.1 Flash main features

Flash memory features:

- Up to 512 Kbytes of flash memory single bank architecture
- Memory organization: 1 bank
  - main memory: 512 Kbytes
  - page size: 2 Kbytes
- 32-bit wide data read
- 32-bit wide data write
- page erase (2 Kbytes) and mass erase

Flash controller features:

- flash memory read operations
- flash memory write operations: single data write or 4x32-bits burst write
- flash memory erase operations
- flash readout protection and disable of debug access
- page write protect mechanism

9.2 Description

The flash memory is organized as follows:

- A main memory block containing 256 pages of 2 Kbytes. Each page is made of 8 rows of 256 bytes (64 words).

Erasing the whole flash memory results in all ones in every bit cell of the flash memory.

In parallel, the flash controller manages 1 Kbyte OTP (One-Time Programmable) for user data. The OTP data cannot be erased. The OTP data area can no more be written only as soon as the last word (address 0x1000_1BFC) is different from 0xFFFF_FFFF and a system reset occurred.

The flash memory is mapped on the AHB-Lite bus with the range described below:

Table 20. Flash memory section address

Section	Flash AHB start address	Flash AHB end address
Main flash	0x1004_0000	0x100B_FFFF (1)
User OTP	0x1000_1800	0x1000_1BFF

1. Depends on flash size. See *Table 22: Flash size information*.

9.3 Flash controller registers

9.3.1 Command register (COMMAND)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMMAND							
								rw							

Bits 31:8 Reserved, must be kept at reset value.

Bit 7:0 **COMMAND**: Command opcode to launch any operation on flash memory. See *Table 21* for command list and detail.

Table 21. Command list available

Command	Flash sector	Description	Command Opcode
ERASE	Main memory	Erase page defined by register ADDRESS.	0x11
MASSERASE	Main memory	Mass erase (main flash memory is completely erased).	0x22
WRITE (1)	Main memory	Program one location (defined by registers DATA and ADDRESS).	0x33
MASSREAD	Main memory	Read all locations and compare with DATA register value or produce LFSR signature.	0x55
SLEEP	Whole memory	Put flash in sleep mode. ***Warning: once this command is launched, no access (read) nor action (any command except WAKE) on the flash component are possible until the WAKE command is requested (and associated CMDDONE status is returned)***	0xAA
WAKEUP	Whole memory	Get flash out of sleep mode.	0xBB
BURSTWRITE	Main memory	Program 4 locations (ADDRESS → ADDRESS+3) with data in DATA0-DATA3 registers. ***Warning: burst always starts (=DATA0 is always written) on a 16-bytes aligned address even if ADDRESS is not 16-bytes aligned (register bit field ADDRESS[1:0] always considered as 0 at flash controller level).***	0xCC
OTPWRITE	User OTP	One time writable 1Kbytes for customer. No erase not second programming is possible.	0xEE
KEYWRITE	Main memory	Write the customer key used to protect the main flash.	0xFF

1. Each address can be programmed only 2 times without erase operation in between.

Status bits behavior versus commands:

- Writing to the COMMAND register starts the action that is performed on the flash.
- The CMDSTART flag goes and stays high until it is cleared.
- When the command has finished the CMDDONE flag goes high.
- When a MASS READ command was given and when CMDDONE is high, the READOK flag can be checked or the LFSRVAL register can be read (contains the signature of the mass read).

The sequences to use the different commands are described in *9.4: Programmer's model on page 190*.

9.3.2 Configuration register (CONFIG)

Address offset: 0x04

Reset value:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAIT_STATES	.	DIS_GROUP_WRITE	REMAP	.
										rw	rw		rw	rw	

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **WAIT_STATES**: Number of wait states to be inserted on flash read (AHB accesses).
The flash embedded in the STM32WB09xE device requires 1 wait_state when system clock frequency is 64 MHz.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DIS_GROUP_WRITE**:
0: burst write operations are allowed/enabled.
1: burst write operations are blocked and result in a single write.
Note: if this bit is set during an on-going burst write operation, the flash controller stops the write operation at the end of the current word writing even if some words are still to be written.

Bit 1 **REMAP**: bit to redirect boot area on SRAM0. See *Table 5: Address remapping depending on the REMAP bit* for details.

Bit 0 Reserved, must be kept at reset value.

The flash can be read in one system clock cycle (the best for power consumption) when the system clock is 32MHz maximum.

9.3.3 Interrupt status register (IRQSTAT)

The interrupt status register shows the masked version of the interrupt raw register.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READOK_MIS	ILLCMD_MIS	CMDERR_MIS	CMDSTART_MIS	CMDDONE_MIS
											rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **READOK_MIS**: Mass read OK masked interrupt status.
This bit is set at the end of a MASSREAD operation if all the words read in the memory match the DATA0 register value.
Cleared by writing 1.

Bit 3 **ILLCMD_MIS**: Illegal command masked interrupt status.
This bit is set when a bad opcode command is written in the COMMAND register.
Cleared by writing 1.

Bit 2 **CMDERR_MIS**: command error masked interrupt status.
This bit is set if a command opcode is written in COMMAND register while the flash is busy.
Cleared by writing 1.

Bit 1 **CMDSTART_MIS**: Command started masked interrupt status.
This bit is set once the requested command execution has started.
Cleared by writing 1.

Bit 0 **CMDDONE_MIS**: Command done masked interrupt status.
This it is set once the requested command execution is completed.
Cleared by writing 1.

The CMDDONE and CMDSTART bits are updated a few clock cycles after the requested command has been started by writing to the COMMAND register.

Note: Clearing a bit by writing in IRQSTAT (respectively IRQRAW) register also cleared the same bit in IRQRAW (respectively IRQSTAT) register as they are referring to a common condition/event.

9.3.4 Interrupt mask register (IRQMASK)

The mask bit in IRQMASK masks the condition in the status register IRQSTAT and prevent the generation of the interrupt.

Address offset: 0x0C

Reset value: 0x0000 003F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ILLCMDM	CMDERRM	CMDSTARTM	CMDDONEM
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ILLCMDM**: Illegal command mask.
- 0: enable interrupt on “illegal command” event.
- 1: disable interrupt on “illegal command” event

Bit 2 **CMDERRM**: command error mask.
- 0: enable interrupt on “command error” event.
- 1: disable interrupt on “command error” event

Bit 1 **CMDSTARTM**: Command started mask.
- 0: enable interrupt on “command started” event.
- 1: disable interrupt on “command started” event

Bit 0 **CMDDONEM**: Command done mask.
- 0: enable interrupt on “command done” event.
- 1: disable interrupt on “command done” event

9.3.5 Raw status register (IRQRAW)

The raw status register shows the unmasked condition of interrupt events.

Address offset: 0x10

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ILLCMD_RIS	CMDERR_RIS	CMDSTART_RIS	CMDDONE_RIS
												rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ILLCMD_RIS**: Illegal command raw/unmasked interrupt status.
This bit is set when a bad opcode command is written in the COMMAND register.
Cleared by writing 1.

Bit 2 **CMDERR_RIS**: command error raw/unmasked interrupt status.
This bit is set if a command opcode is written in COMMAND register while the flash is busy.
Cleared by writing 1.

Bit 1 **CMDSTART_RIS**: Command started raw/unmasked interrupt status.
This bit is set once the requested command execution has started.
Cleared by writing 1.

Bit 0 **CMDDONE_RIS**: Command done raw/unmasked interrupt status.
This it is set once the requested command execution is completed.
Cleared by writing 1.

The CMDDONE and CMDSTART bits are updated a few clock cycles after the requested command has been started by writing to the COMMAND register.

Note: Clearing a bit by writing in IRQSTAT (respectively IRQRAW) register also cleared the same bit in IRQRAW (respectively IRQSTAT) register as they are referring to a common condition/event.

9.3.6 SIZE register (SIZE)

Address offset: 0x14

Reset value: 0x000 XXXX (depends on device)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWD_DISABLE	FLASH_SECURE	RAM_SIZE[1:0]		FLASH_SIZE[16]
												-	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE[15:0]															
r															

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **SWD_DISABLE**: Flash +SWD protection:
0: No SWD protection (refer to FLASH_SECURE)
1: FLASH and SWD protected

Bit 19 **FLASH_SECURE**: indicates the main FLASH is locked by a customer key.
0: the main FLASH is not protected.
1: the main FLASH is protected through a customer key.

Bits 18:17 **RAM_SIZE**: indicates the size of RAM available in the device:
00: 32 Kbytes of RAM available (RAM0 and RAM1 banks)
01: 32 Kbytes of RAM available (RAM0 and RAM1 banks)
10: 48 Kbytes of RAM available (RAM0, RAM1and RAM2 banks)
11: 64 Kbytes of RAM available (RAM0, RAM1, RAM2 and RAM3 banks)

Bits 16:0 **FLASH_SIZE**: indicates the last usable address of the flash using memory component address format. See *Table 22* for relation between address at flash component level and AHB address mapping.
0xBFFF: 192 Kbytes of main flash are available on this device.
0xFFFF: 256 Kbytes of main flash are available on this device.
0x17FFF: 384 Kbytes of main flash are available on this device.
0x1FFFF: 512 Kbytes of main flash are available on this device.

Table 22. Flash size information

Main flash size	Highest usable address at flash level(1)	Highest usable address at AHB level
192 Kbytes	0xBFFF	0x1006_FFFC
256 Kbytes	0xFFFF	0x1007_FFFC
384 Kbytes	0x17FFF	0x1009_FFFC
512 Kbytes	0x1FFFF	0x100B_FFFC

1. Value seen in FLASH_SIZE bit field.

9.3.7 Address register (ADDRESS)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XADDR [10]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XADDR[9:0]										YADDR[5:0]					
rw										rw					

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:6 **XADDR[10:0]**:
- XADDR[10:3]: page number (from 0 to 255)
- XADRR[2:0]: row number (from 0 to 7)

Bits 5:0 **YADDR[5:0]**: word number inside the selected row (from 0 to 63)

Address to provide to the flash is not the AHB device mapping address but the address respecting flash component format.

The main flash is composed of 256 pages containing 8 rows each with 64 words = 256 bytes by row.

To program the ADDRESS register, the formula is the following:
- XADDR[10:0] = AHB address bit[18:8]
- YADDR[5:0] = AHB address bit[7:2]

Example 1: To program a word (32-bit) at AHB address 0x1005_0454:
- XADDR[10:0] = AHB address bit[18:8] = 0x104
- YADDR[5:0] = AHB address bit[7:2] = 0x15
- ADDRESS register = 0x4115

9.3.8 Linear Feedback Shift register (LFSRVAL)

the LFSRVAL register contains the signature issued by a MASSREAD command.

The LFSRVAL register is initialized with all ones when the MASS READ command is written to the COMMAND register. Then every read value is put through the LFSR.

The final signature can be read in this register once the CMDDONE information is set.

customer.

Address offset: 0x24

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LFSRVAL[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LFSRVAL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **LSFRVAL**: signature after a MASSREAD command, generated through a Linear Feedback Shift Register block.

9.3.9 Main flash page protection registers (PAGEPROTx)

The PAGEPROTx registers allows protecting from accidental write a contiguous set of pages called segment in the following description. A maximum of four segments can be defined.

An example of usage is available in the

PAGEPROT0

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEG1[15:0]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEG0[15:0]															
rw															

Bits 31:16 **SEG1**: second segment definition.
See SEG0 description for details on SEG1[15:0] content.

Bits 15:0 **SEG0**: First segment definition.
A segment SEGx is built as follows:
– SEGx[15:8] = OFFSET: page number to start the write protection (value between 0 and 0xFF)
– SEGx[7:0] = SIZE: number of 2 Kbyte pages to protect including the starting page (provided in SEGx[15:8]).

Notes:

- SIZE=0 means no segment is defined, so if all segments have SIZE=0, then no write protection is applied on the whole flash memory.
- The segments can overlap, the protection on a page is guaranteed if at least one segment covers this page.
- If OFFSET + SIZE > 255d, and so exceeds the maximum size of the flash memory, the end of the segment is positioned on the maximum allowed address.

PAGEPROT1

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEG3[15:0]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEG2[15:0]															
rw															

Bits 31:16 **SEG3**: fourth segment definition.
See PAGEPROT0.SEG0 description for details on SEG3[15:0] content.

Bits 15:0 **SEG2**: third segment definition.
See PAGEPROT0.SEG0 description for details on SEG2[15:0] content.

9.3.10 Data registers (DATA0-DATA3)

The DATA0 register needs to be written with:

- The desired value written to the flash location (for single write or mass write).
- The desired compare value for a (mass) read operation, the flag READOK indicates if there was a match or not. For mass read, all read values must match for READOK.

The DATA1-DATA3 registers need to be written only for burst write.

DATA0

Address offset: 0x40

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA0[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0[15:0]															
rw															

Bits 31:0 **DATA0**: this register has several usages:
- Data to write in flash in single write mode
- First data to be written in flash on a burst write
- Compared value for a MASSREAD command (useful only if flash is fully written with the same word)

Note: In this last case, the flag READOK indicates if there was a match or not at the end of the mass read.

DATA1

Address offset: 0x44

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA1[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[15:0]															
rw															

Bits 31:0 **DATA1**: data that is written at ADDRESS+1 during a BURSTWRITE command.
Note: this register is used only on burst write.

DATA2

Address offset: 0x48

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA2[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA2[15:0]															
rw															

Bits 31:0 **DATA2**: data that is written at ADDRESS+2 during a BURSTWRITE command.
Note: this register is used only on burst write.

DATA3

Address offset: 0x4C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA3[15:0]															
rw															

Bits 31:0 **DATA3**: data that is written at ADDRESS+3 during a BURSTWRITE command.
Note: this register is used only on burst write.

9.3.11 Flash controller register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the flash controller base address location in the STM32WB09xE.

Table 23. Flash controller register map and reset values

Offset	Register	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
0x00	**COMMAND**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMMAND[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x04	**CONFIG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAIT_STATE[1:0]		Res.	DIS_GROUP_WRITE	REMAP	Res.
	Reset value																											0	1		0	0	1
0x08	**IRQSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READOK_MIS	ILLCMD_MIS	CMDERR_MIS	CMDSTART_MIS	CMDDONE_MIS
	Reset value																												0		0	0	0
0x0C	**IRQMASK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ILLCMDM	CMDERRM	CMDSTARTM	CMDDONEM
	Reset value																													1	1	1	1
0x10	**IRQRAW**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ILLCMD_RIS	CMDERR_RIS	CMDSTART_RIS	CMDDONE_RIS
	Reset value																														0	0	0
0x14	**SIZE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWD_DISABLE	FLASH_SECURE	RAM_SIZE[1:0]		FLASH_SIZE[16:0]																	
	Reset value													-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**ADDRESS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XADDR[10:0]													YADDR[5:0]					
	Reset value																0	0		0		0		0		0		0		0		0		
0x24	**LFSRVAL**	LSFRVAL[31:0]																															
	Reset value	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

Table 23. Flash controller register map and reset values (continued)

Offset	Register	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
0x34	**PAGEPROT0**	SEG1[15:0]																SEG0[15:0]															
	Reset value	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
0x38	**PAGEPROT1**	SEG3[15:0]																SEG2[15:0]															
	Reset value	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
0x3C	Reserved																																
0x40	**DATA0**	DATA0[31:0]																															
	Reset value	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
0x44	**DATA1**	DATA1[31:0]																															
	Reset value	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
0x48	**DATA2**	DATA2[31:0]																															
	Reset value	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
0x4C	**DATA3**	DATA3[31:0]																															
	Reset value	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

9.4 Programmer's model

The STM32WB09xE embeds up to 512 Kbytes of internal flash memory. A flash interface implements instruction access and data access based on the AHB protocol. It implements the logic necessary to carry out the flash memory operations (Program/Erase) controlled through the flash registers.

9.4.1 General information

Writing to flash memory only allows clearing bits from '1' to '0'. This means any write from '0' to '1' implies erasing before performing a write.

Flash memory is composed of 256 pages containing 8 rows of 64 words (256 x 8 x 64 = 131072 words). Each word is 32-bit = 4 bytes long which means 512kB of flash.

The address inside the flash controller ADDRESS register is built as follows:

ADDRESS[15:0] = XADR[10:0] & YADR[5:0] with

- XADR[10:3] = page address
- XADR[2:0] = row address
- YADR[5:0] = word address (one word = four bytes)

Note: One specific address can be written only twice between two erase actions even if each writing only clears bits.

9.4.2 Read function example

There are two possible read accesses:

- Read one single word: simple read as if SRAM: read the desired flash address and get read data on the bus.
- MASSREAD command: read the full flash memory and compare with expected content.

There are two ways of using MASSREAD:

- Full flash contains a fixed 32-bit pattern: indicate the expected pattern (value to be compared with each read value inside flash) in the flash controller DATA register and check the READOK flag in the flash controller Interrupt register once the command is completed.
- Else: request a MASSREAD command without specifying any expected read value and check the LFSRVAL register once the command is completed. This LFSRVAL register contains a signature of the memory read.

MASSREAD sequence:

1. Write in the flash controller DATA register the expected value (if MASSREAD is used in combination with the READOK flag).
2. Write the MASSREAD command (0x55) in the flash controller COMMAND register.
3. Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
4. Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
5. Then, wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
6. Check the READOK flag (expected high) in the IRQSTAT register or the LFSRVAL register value to ensure flash memory content is the expected result.
7. Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register.

9.4.3 Erase function example

The flash controller allows the erasure of one page, or full erasure of the main flash-memory.

ERASE sequence (erase one page):

1. Write the page address to be erased by writing in the flash controller ADDRESS register the following value:
2. ADDRESS[16:9] = XADR[10:3] = page address to erase
3. ADDRESS[8:0] = 9'b0 (row and word addresses at zero).
4. Write the ERASE command (0x11) in the flash controller COMMAND register.
5. Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating command is taken into account and under execution.
6. Clear the CMDSTART flag by writing CMDSTART to '1' in the flash controller IRQSTAT register.
7. Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
8. Clear the CMDDONE flag by writing CMDDONE to '1' in the flash controller IRQSTAT register.
9. After this command, the erased page contains only bits set to '1'.

MASSERASE sequence (erase whole main flash):

1. Write the MASSERASE command (0x22) in the flash controller COMMAND register.
2. Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
3. Clear the CMDSTART flag by writing CMDSTART to '1' in the flash controller IRQSTAT register.
4. Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
5. Clear the CMDDONE flag by writing CMDDONE to '1' in the flash controller IRQSTAT register.
6. After this command, the full main flash contains only bits set to '1'.

9.4.4 Write function example

The flash controller allows writing one word (WRITE), up to 4 words (BURSWRITE) or the full main flash memory (with a single fixed word).

Note: As a write can only program to '0' on bits already set to '1', it is necessary to erase the page and request that the bits be set to '1' (instead of '0') in order to re-write to '0'.

WRITE sequence:

1. Indicate the location to write by filling the flash controller ADDRESS register with the targeted address (page, row and word number)
2. Write the value to program in the flash controller DATA register.
3. Write the WRITE command (0x33) in the flash controller COMMAND register.
4. Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
5. Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
6. Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
7. Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register.

BUSRTWRITE sequence:

1. Indicate the location to write by filling the flash controller ADDRESS register with the targeted address of the first data to write (page, row and word number). DATA0 is written at ADDRESS, DATA1 at ADDRESS+1, etc.
2. Write the values to program in the flash controller DATA0-3 registers. To write less than four words, write 0xFFFFFFFF in the unused DATA1-3 registers.
3. Write the BURSTWRITE command (0xCC) in the flash controller COMMAND register.
4. Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
5. Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
6. Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command is completed.
7. Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register.

9.4.5 Enabling protection example

The device offers three level of protection to prevent application cloning and or altering of application code. The different levels are described in the *Table 24*. It is important to notice that disabling of SWD access is an irreversible operation.

Table 24. System memory protection

Protection	UserKey[Data0]	UserKey[Data1]	Disable protection	Description
None	0xFFFFFFFF	0xFFFFFFFF	Not applicable	Debugger can access and modify flash memory and RAM content. This is the default configuration. All the bootloader commands are available.
Readout	0xAAAAAAAA	0xAAAAAAAA	Perform mass erase	Debugger cannot read or modify both flash memory and RAM content. The bootloader commands READ_MEMORY, WRITE_MEMORY and GO are disabled and MASS ERASE is available.
SWD	0xABACABAD	0xABACABAD	This selection is irreversible	Debugger connection is not possible. There is no possibility to access to the device via SWD and all the bootloader commands are disabled.
Not specified	Any other value	Any other value	Not applicable	This configuration is forbidden and can lead to unrecoverable damage of the device

In order to activate the desired level of protection, the following KEYWRITE sequence should be used:

1. Write DATA0 (LSB key) and DATA1 (MSB key) registers with the value to program
2. Write 0xC7EF584D to DATA2 and 0xB3A21096 to DATA3
3. Write the KEYWRITE command (0xFF) in the flash controller COMMAND register
4. Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
5. Clear the CMDSTART flag by writing CMDSTART to ‘1’ in the flash controller IRQSTAT register.
6. Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode), indicating that the command is completed.
7. Clear the CMDDONE flag by writing CMDDONE to ‘1’ in the flash controller IRQSTAT register. The key are activated (and flash and RAM banks are protected) after a reset.

Note: The secure bootloader capability offers a further security level for preventing unwanted control from external users.
When the secure bootloader capability is enabled the flash and RAM contents are readable. The GO command of the bootloader is disabled and all the other bootloader commands are available. Only signed firmware can be executed.

For more information about the secure bootloader capability refer to application note AN6140: How to use the secure bootloader on STM32WB0 MCUs.

9.4.6 OTP function example

OTPWRITE sequence:

1. Write DATA0 register with the value to program (no burst write feature is available as only few bytes to be written once only)
2. Write ADDRESS register according to the following rule:
   ADDRESS[16:9] = don't care (page number frozen by hardware on this command)
   ADDRESS[8:6] = 0, 1, 2 or 3
   ADDRESS[5:0] = full area possible (from 0x00 to 0x3F)
3. Write the OTPWRITE command (0xEE) in the flash controller COMMAND register
4. Wait for the CMDSTART flag in the IRQSTAT register (polling mode or interrupt mode) indicating that the command has been taken into account and is under execution.
5. Clear the CMDSTART flag by writing CMDSTART to '1' in the flash controller IRQSTAT register.
6. Wait for the CMDDONE flag in the IRQSTAT register (polling mode or interrupt mode), indicating that the command is completed.
7. Clear the CMDDONE flag by writing CMDDONE to '1' in the flash controller IRQSTAT register.

Note: The OTP locations are following flash memory rules, that is, a second write only flip bit from 1 to 0. If the user wishes to lock the OTP values and prevent any further modification in the OTP area, it must write the last OTP word (address 0x10001BFC) with a value different from 0xFFFFFFFF and perform system reset. The operation of locking the OTP area is irreversible.

9.4.7 Write page protection example

Example to write protect against accidental programming knowing the flash starts at address 0x1004_0000 and contains 256 pages of 2 Kbytes (pages 0 to 255).

- the address ranges 0x1004C000-0x1004FFFF

Starting page: 0xC000 / 0x800 = 0x18

SEG0[15:8]= 0x18 (OFFSET = 0x18)

Number of pages: (0x10000 - 0xC000) / 0x800 = 0x8

SEG0[7:0]= 0x08 (SIZE = 0x08)

- and the address ranges 0x1005E000-0x1005FFFF.

Starting page: 0x1E000 / 0x800 = 0x3C

SEG1[15:8] = 0x3C (OFFSET = 0x3C)

Number of pages: (0x20000 - 0x1E000) / 0x800 = 0x4

SEG1[7:0] = 0x04 (SIZE = 0x04)

Conclusion: Program the PAGEPROT0 = 0x3C041808.

10 DMA controller (DMA)

10.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The DMA has an arbiter for handling the priority between DMA requests.

10.2 DMA main features

- Eight independently configurable channels (requests)
- Each of the eight channels is connected to dedicated hardware DMA requests, software trigger is also supported on each channel. This configuration is done by software.
- Priorities between requests from channels of the DMA are software programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (request 1 has priority over request 2, etc.)
- Independent source and destination transfer size (byte, half word, word), emulating packing and unpacking. Source/destination addresses must be aligned on the data size.
- Support for circular buffer management
- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) logically ORed together in a single interrupt request for each channel
- Memory-to-memory transfer (SRAM0/SRAM1).
- Peripheral-to-memory and memory-to-peripheral, and peripheral-to-peripheral transfers
- Access to SRAMs, APB0 and APB1 peripherals as source and destination
- Programmable number of data to be transferred: up to 65536

10.3 DMA functional description

The DMA controller performs direct memory transfer by sharing the system bus with the other masters of the device. The DMA request may stop the CPU access to the system bus for some bus cycles, when the CPU and DMA are targeting the same destination (memory or peripheral). The bus matrix implements round-robin scheduling, thus ensuring at least half of the system bus bandwidth (both to memory and peripheral) for the CPU.

10.3.1 DMA transactions

After an event, the peripheral sends a request signal to the DMA Controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA Controller accesses the peripheral, an Acknowledge is sent to the peripheral by the DMA Controller. The peripheral releases its request as soon as it gets the Acknowledge from the DMA Controller. Once the request is deasserted by the peripheral, the DMA Controller

release the Acknowledge. If there are more requests, the peripheral can initiate the next transaction.

In summary, each DMA transfer consists of three operations:

- The loading of data from the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The storage of the data loaded to the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The post-decrementing of the DMA_CNDTRx register, which contains the number of transactions that have still to be performed.

10.3.2 Arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences.

The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA_CCRx register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lowest number gets priority versus the channel with the highest number. For example, channel 2 gets priority over channel 4.

10.3.3 DMA channels

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred (up to 65535) is programmable. The register which contains the amount of data items to be transferred is decremented after each transaction.

Programmable data sizes

Transfer data sizes of the peripheral and memory are fully programmable through the PSIZE and MSIZE bits in the DMA_CCRx register.

Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented after each transaction depending on the PINC and MINC bits in the DMA_CCRx register. If incremented mode is enabled, the address of the next transfer is the address of the previous one incremented by 1, 2 or 4 depending on the chosen data size. The first transfer address is the one programmed in the DMA_CPARx/DMA_CMARx registers. During transfer operations, these registers keep the initially programmed value. The current

transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel is configured in noncircular mode, no DMA request is served after the last transfer (that is once the number of data items to be transferred has reached zero). In order to reload a new number of data items to be transferred into the DMA_CNDTRx register, the DMA channel must be disabled.

Note: If a DMA channel is disabled, the DMA registers are not reset. The DMA channel registers (DMA_CCRx, DMA_CPARx and DMA_CMARx) retain the initial values programmed during the channel configuration phase.

In circular mode, after the last transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address vaules from the DMA_CPARx/DMA_CMARx registers.

Channel configuration procedure

The following sequence should be followed to configure a DMA channelx (where x is the channel number).

1. Set the peripheral register address in the DMA_CPARx register. The data is moved from/ to this address to/ from the memory after the peripheral event.
2. Set the memory address in the DMA_CMARx register. The data is written to or read from this memory after the peripheral event.
3. Configure the total number of data to be transferred in the DMA_CNDTRx register. After each peripheral event, this value is decremented.
4. Configure the channel priority using the PL[1:0] bits in the DMA_CCRx register
5. Configure data transfer direction, circular mode, peripheral and memory incremented mode, peripheral and memory data size, and interrupt after half and/or full transfer in the DMA_CCRx register
6. Activate the channel by setting the ENABLE bit in the DMA_CCRx register.

As soon as the channel is enabled, it can serve any DMA request from the peripheral connected on the channel.

Once half of the bytes are transferred, the half-transfer flag (HTIF) is set and an interrupt is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. At the end of the transfer, the Transfer Complete Flag (TCIF) is set and an interrupt is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.

Circular mode

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_CCRx register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This mode is called Memory to Memory mode.

If the MEM2MEM bit in the DMA_CCRx register is set, then the channel initiates transfers as soon as it is enabled by software by setting the Enable bit (EN) in the DMA_CCRx

register. The transfer stops once the DMA_CNDTRx register reaches zero. Memory to Memory mode may not be used at the same time as Circular mode.

10.3.4 Programmable data width, data alignment and endians

When PSIZE and MSIZE are not equal, the DMA performs some data alignments as described in *Table 25: Programmable data width and endian behavior (when PINC=MINC=1 and NDT=4)*.

Table 25. Programmable data width and endian behavior (when PINC=MINC=1 and NDT(1)=4)

Port width	Source content	Transfer operation	Dest content
Src => Dest	Addr / data		Addr / data
8 => 8	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	Read B0[7:0] @0x0 then Write B0[7:0] @0x0 Read B1[7:0] @0x1 then Write B1[7:0] @0x1 Read B2[7:0] @0x2 then Write B2[7:0] @0x2 Read B3[7:0] @0x3 then Write B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8 => 16	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	Read B0[7:0] @0x0 then Write 00B0[15:0] @0x0 Read B1[7:0] @0x1 then Write 00B1[15:0] @0x2 Read B2[7:0] @0x2 then Write 00B2[15:0] @0x4 Read B3[7:0] @0x3 then Write 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8 => 32	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	Read B0[7:0] @0x0 then Write 000000B0[31:0] @0x0 Read B1[7:0] @0x1 then Write 000000B1[31:0] @0x4 Read B2[7:0] @0x2 then Write 000000B2[31:0] @0x8 Read B3[7:0] @0x3 then Write 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16 => 8	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4/ @0x6 / B7B6	Read B1B0[15:0] @0x0 then Write B0[7:0] @0x0 Read B3B2[15:0] @0x2 then Write B2[7:0] @0x1 Read B5B4[15:0] @0x4 then Write B4[7:0] @0x2 Read B7B6[15:0] @0x6 then Write B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16 => 16	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4/ @0x6 / B7B6	Read B1B0[15:0] @0x0 then Write B1B0[15:0] @0x0 Read B3B2[15:0] @0x2 then Write B3B2[15:0] @0x2 Read B5B4[15:0] @0x4 then Write B5B4[15:0] @0x4 Read B7B6[15:0] @0x6 then Write B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16 => 32	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4/ @0x6 / B7B6	Read B1B0[15:0] @0x0 then Write 0000B1B0[31:0] @0x0 Read B3B2[15:0] @0x2 then Write 0000B3B2[31:0] @0x4 Read B5B4[15:0] @0x4 then Write 0000B5B4[31:0] @0x8 Read B7B6[15:0] @0x6 then Write 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32 => 8	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC/BFBEBDBC	Read B3B2B1B0[31:0] @0x0 then Write B0[7:0] @0x0 Read B7B6B5B4[31:0] @0x4 then Write B4[7:0] @0x1 Read BBBAB9B8[31:0] @0x8 then Write B8[7:0] @0x2 Read BFBEBDBC[31:0] @0xC then Write BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC

Table 25. Programmable data width and endian behavior (when PINC=MINC=1 and NDT(1)=4)

Port width	Source content	Transfer operation	Dest content
Src => Dest	Addr / data		Addr / data
32 => 16	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8/ BBBAB9B8 @0xC/BFBEBDBC	Read B3B2B1B0[31:0] @0x0 then Write B1B0[15:0] @0x0 Read B7B6B5B4[31:0] @0x4 then Write B5B4[15:0] @0x2 Read BBBAB9B8[31:0] @0x8 then Write B9B8[15:0] @0x4 Read BFBEBDBC[31:0] @0xC then Write BDBC[15:0] @0x6	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32 => 32	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 /BBBAB9B8 @0xC/BFBEBDBC	Read B3B2B1B0[31:0] @0x0 then Write B3B2B1B0[31:0] @0x0 Read B7B6B5B4[31:0] @0x4 then Write B7B6B5B4[31:0] @0x4 Read BBBAB9B8[31:0] @0x8 then Write BBBAB9B8[31:0] @0x8 Read BFBEBDBC[31:0] @0xC then Write BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 /BBBAB9B8 @0xC/BFBEBDBC

1. NDT = Number of data items to transfer.

Addressing an AHB peripheral that does not support byte or halfword write operations

When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. So when the used AHB slave peripheral does not support byte or halfword write operations (when HSIZE is not used by the peripheral) and does not generate any error, the DMA writes the 32 HWDATA bits as shown in the two examples below:

- To write the halfword “0xABCD”, the DMA sets the HWDATA bus to “0xABCDABCD” with HSIZE = HalfWord
- To write the byte “0xAB”, the DMA sets the HWDATA bus to “0xABABABAB” with HSIZE = Byte

Assuming that the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it transforms any AHB byte or halfword operation into a 32-bit APB operation in the following manner:

- an AHB byte write operation of the data “0xB0” to 0x0 (or to 0x1, 0x2 or 0x3) is converted to an APB word write operation of the data “0xB0B0B0B0” to 0x0
- an AHB halfword write operation of the data “0xB1B0” to 0x0 (or to 0x2) is converted to an APB word write operation of the data “0xB1B0B1B0” to 0x0

For instance, if you want to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), you must configure the memory source size (MSIZE) to “16-bit” and the peripheral destination size (PSIZE) to “32-bit”.

10.3.5 Error management

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding

Channel configuration register (DMA_CCRx). The channel's transfer error interrupt flag (TEIF) in the DMA_IFR register is set and an interrupt is generated if the transfer error interrupt enable bit (TEIE) in the DMA_CCRx register is set.

10.3.6 Interrupts

An interrupt can be produced on a half-transfer, transfer complete or transfer error for each DMA channel. Separate interrupt enable bits are available for flexibility.

Table 26. DMA interrupt requests

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

10.3.7 DMA request mapping

A DMAMUX is present in the STM32WB09xE device and allows selecting which requester is connected to which DMA channel. See *Table 29: DMAMUX: assignment of multiplexer inputs to resources* for details.

10.4 DMA registers

Refer to *Section 1.5: Acronyms* for a list of abbreviations used in register descriptions.

The peripheral registers must be accessed by words (32-bit) only.

10.4.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEIF8	HTIF8	TCIF8	GIF8	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 31, 27, 23, 19, 15, 11, 7, 3 **TEIFx:** Channel x transfer error flag (x = 1..8)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.
0: No transfer error (TE) on channel x
1: A transfer error (TE) occurred on channel x

Bits 30, 26, 22, 18, 14, 10, 6, 2 **HTIFx:** Channel x half transfer flag (x = 1..8)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.
0: No half transfer (HT) event on channel x
1: A half transfer (HT) event occurred on channel x

Bits 29, 25, 21, 17, 13, 9, 5, 1 **TCIFx:** Channel x transfer complete flag (x = 1..8)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.
0: No transfer complete (TC) event on channel x
1: A transfer complete (TC) event occurred on channel x

Bits 28, 24, 20, 16, 12, 8, 4, 0 **GIFx:** Channel x global interrupt flag (x = 1..8)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.
0: No TE, HT or TC event on channel x
1: A TE, HT or TC event occurred on channel x

10.4.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTEIF 8	CHTIF 8	CTCIF 8	CGIF8	CTEIF 7	CHTIF 7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF 4	CHTIF 4	CTCIF 4	CGIF4	CTEIF 3	CHTIF 3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:28 Reserved, must be kept at reset value.

Bits 31, 27, 23, 19, 15, 11, 7, 3 **CTEIFx:** Channel x transfer error clear (x = 1..8)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding TEIF flag in the DMA_ISR register

Bits 30, 26, 22, 18, 14, 10, 6, 2 **CHTIFx:** Channel x half transfer clear (x = 1..8)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding HTIF flag in the DMA_ISR register

Bits 29, 25, 21, 17, 13, 9, 5, 1 **CTCIFx:** Channel x transfer complete clear (x = 1..8)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding TCIF flag in the DMA_ISR register

Bits 28, 24, 20, 16, 12, 8, 4, 0 **CGIFx:** Channel x global interrupt clear (x = 1..8)
This bit is set and cleared by software.
0: No effect
1: Clears the GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register

10.4.3 DMA channel x configuration register (DMA_CCRx) (x = 1..8, where x = channel number)

Address offset: 0x008 + 0d20 × (channel number - 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **MEM2MEM:** Memory to memory mode
This bit is set and cleared by software.
0: Memory to memory mode disabled
1: Memory to memory mode enabled

Bits 13:12 **PL[1:0]:** Channel priority level
These bits are set and cleared by software.
00: Low
01: Medium
10: High
11: Very high

Bits 11:10 **MSIZE[1:0]:** Memory size
These bits are set and cleared by software.
00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bits 9:8 **PSIZE[1:0]:** Peripheral size
These bits are set and cleared by software.
00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bit 7 **MINC:** Memory increment mode
This bit is set and cleared by software.
0: Memory increment mode disabled
1: Memory increment mode enabled

Bit 6 **PINC:** Peripheral increment mode
This bit is set and cleared by software.
0: Peripheral increment mode disabled
1: Peripheral increment mode enabled

Bit 5 **CIRC:** Circular mode
This bit is set and cleared by software.
0: Circular mode disabled
1: Circular mode enabled

Bit 4 **DIR:** Data transfer direction
This bit is set and cleared by software.
0: Read from peripheral
1: Read from memory

Bit 3 **TEIE:** Transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 2 **HTIE:** Half transfer interrupt enable
This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled

Bit 1 **TCIE:** Transfer complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 0 **EN:** Channel enable
This bit is set and cleared by software.
0: Channel disabled
1: Channel enabled

10.4.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1..8, where x = channel number)

Address offset: 0x00C + 0d20 × (channel number - 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]:** Number of data to transfer

Number of data to be transferred (0 up to 65535). This register can only be written when the channel is disabled. Once the channel is enabled, this register is read-only, indicating the remaining bytes to be transmitted. This register decrements after each DMA transfer.

Once the transfer is completed, this register can either stay at zero or be reloaded automatically by the value previously programmed if the channel is configured in auto-reload mode.

If this register is zero, no transaction can be served whether the channel is enabled or not.

10.4.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1..8, where x = channel number)

Address offset: 0x010 + 0d20 × (channel number - 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PA[31:0]:** Peripheral address

Base address of the peripheral data register from/to which the data is read/written.

When PSIZE is 01 (16-bit), the PA[0] bit is ignored. Access is automatically aligned to a half-word address.

When PSIZE is 10 (32-bit), PA[1:0] are ignored. Access is automatically aligned to a word address.

10.4.6 DMA channel x memory address register (DMA_CMARx) (x = 1..8, where x = channel number)

Address offset: 0x014 + 0d20 × (channel number - 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MA[31:0]:** Memory address

Base address of the memory area from/to which the data is read/written.

When MSIZE is 01 (16-bit), the MA[0] bit is ignored. Access is automatically aligned to a half-word address.

When MSIZE is 10 (32-bit), MA[1:0] are ignored. Access is automatically aligned to a word address.

10.4.7 DMA register map

The following table gives the DMA register map and the reset values.

Table 27. DMA register map and reset values

Offset	Register	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
0x000	**DMA_ISR**	TEIF8	HTIF8	TCIF8	GIF8	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value	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
0x004	**DMA_IFCR**	CTEIF8	CHTIF8	CTCIF8	CGIF8	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Reset value	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
0x008	**DMA_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**DMA_CNDTR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**DMA_CPAR1**	PA[31:0]																															
	Reset value	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
0x014	**DMA_CMAR1**	MA[31:0]																															
	Reset value	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
0x01C	**DMA_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value															0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**DMA_CNDTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**DMA_CPAR2**	PA[31:0]																															
	Reset value	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
0x028	**DMA_CMAR2**	MA[31:0]																															
	Reset value	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
0x030	**DMA_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value															0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	**DMA_CNDTR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**DMA_CPAR3**	PA[31:0]																															
	Reset value	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
0x03C	**DMA_CMAR3**	MA[31:0]																															
	Reset value	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
0x044	**DMA_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value															0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27. DMA register map and reset values

Offset	Register	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
0x048	**DMA_CNDTR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	**DMA_CPAR4**	PA[31:0]																															
	Reset value	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
0x050	**DMA_CMAR4**	MA[31:0]																															
	Reset value	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
0x054	Reserved																																
0x058	**DMA_CCR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	**DMA_CNDTR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	**DMA_CPAR5**	PA[31:0]																															
	Reset value	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
0x064	**DMA_CMAR5**	MA[31:0]																															
	Reset value	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
0x068	Reserved																																
0x06C	**DMA_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value															0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	**DMA_CNDTR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	**DMA_CPAR6**	PA[31:0]																															
	Reset value	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
0x078	**DMA_CMAR6**	MA[31:0]																															
	Reset value	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
0x07C	Reserved																																
0x080	DMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value															0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**DMA_CNDTR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	**DMA_CPAR7**	PA[31:0]																															
	Reset value	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
0x08C	**DMA_CMAR7**	MA[31:0]																															
	Reset value	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
0x090	Reserved																																
0x094	DMA_CCR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value															0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27. DMA register map and reset values

Offset	Register	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
0x088	**DMA_CNDTR8**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C	**DMA_CPAR8**	PA[31:0]																															
	Reset value	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
0x0A0	**DMA_CMAR8**	MA[31:0]																															
	Reset value	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
x0A4 to 0x31C	Reserved																																

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

11 DMA request multiplexer (DMAMUX)

11.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until it is served by the DMA controller which generates a DMA acknowledge signal and the corresponding DMA request signal is de-asserted.

For simplicity, the functional description of the DMA request/acknowledge protocol and its associated control signals are abstracted in this document and globally named as DMA request lines. The DMA controller response signals are not shown in figures nor described in the text.

The DMAMUX request multiplexer allows routing a DMA request line between the peripherals and the DMA controller of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line.

11.2 DMAMUX main features

- 8-channel programmable DMA request line multiplexer output.
- Per DMA request line multiplexer channel output:
  - 25 input DMA request lines from peripherals.
  - One DMA request line output.

11.3 DMAMUX implementation

11.3.1 DMAMUX instantiation

DMAMUX is instantiated with the following hardware configuration parameters.

Table 28. DMAMUX instantiation

Feature	DMAMUX
Number of DMAMUX output request channels	8
Number of DMAMUX request generator channels	1
Number of DMAMUX request trigger inputs	2
Number of DMAMUX synchronization inputs	2
Number of DMAMUX peripheral request inputs	25

11.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

Table 29. DMAMUX: assignment of multiplexer inputs to resources

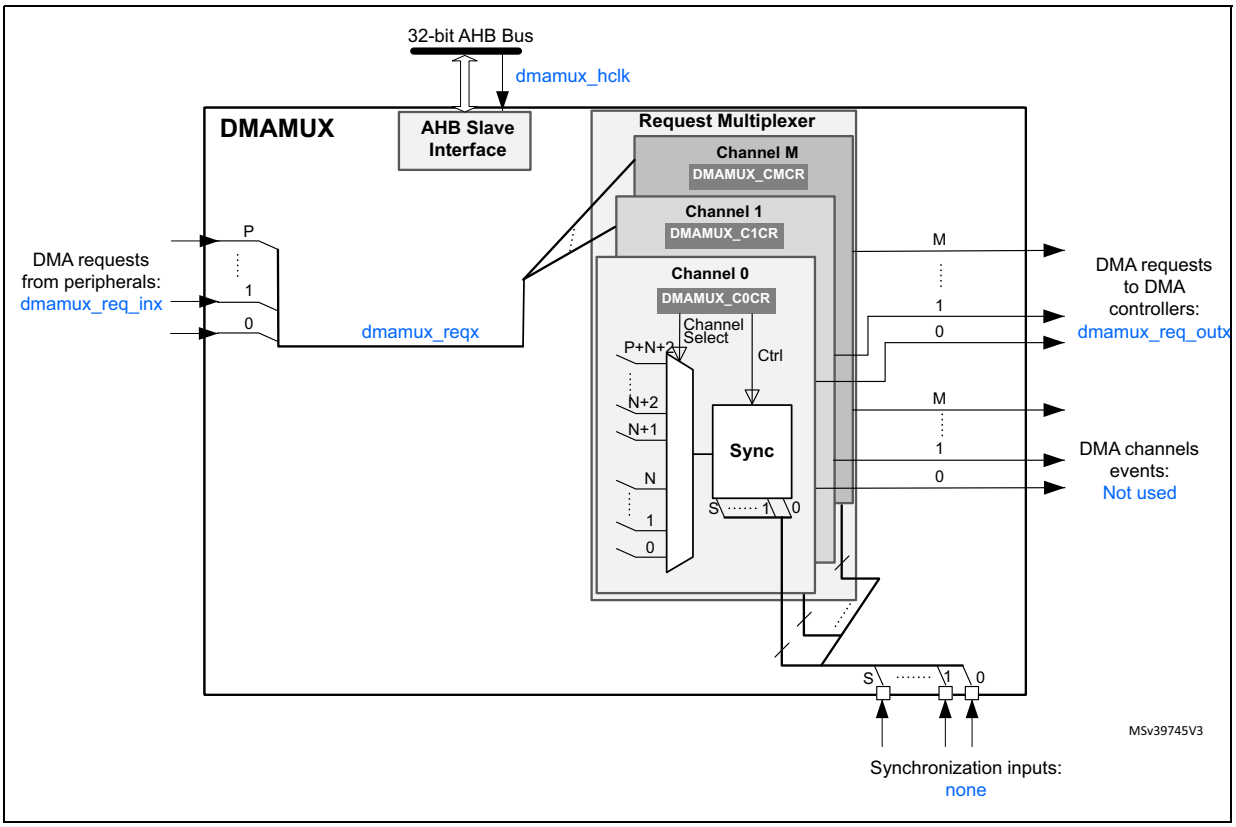
DMA request MUX input	Resource	DMA request MUX input	Resource
1	Reserved	14	LPUART_RX
2	SPI3_RX	15	LPUART_TX
3	SPI3_TX	16	ADC_CH0 (DS channel)
4	Reserved	17	Reserved
5	Reserved	18	TIM2_CH1
6	Reserved	19	TIM2_CH2
7	Reserved	20	TIM2_CH3
8	I2C1_RX	21	TIM2_CH4
9	I2C1_TX	22	TIM2_UP
10	Reserved	23	TIM16_CH1
11	Reserved	24	TIM16_UP
12	USART_RX	25	TIM17_CH1
13	USART_TX	26	TIM17_UP

11.4 DMAMUX functional description

11.4.1 DMAMUX block diagram

Figure 21 shows the DMAMUX block diagram.

Figure 21. DMAMUX block diagram



The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) from .
- DMAMUX requests outputs to channels of DMA controllers (dmamux_req_outx).

11.4.2 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel which may include, depending on the selected input of the request multiplexer.

A DMAMUX request multiplexer channel is connected and dedicated to one single DMA controller(s) channel.

Channel configuration procedure

The following sequence should be followed to configure both a DMAMUX x channel and the related DMA channel y:

11.4.3 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named as DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced from the peripherals .

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the 8-bit DMAREQ_ID field in the DMAMUX_CxCR register.

Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected.

A same non-null DMA_REQ_ID value shall not be programmed to different x and y DMAMUX request multiplexer channels (via DMAMUX1_CxCR and DMAMUX CyCR).

It is not allowed to configure a same non-null DMAREQ_ID to two different channels of the DMAMUX request line multiplexer.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

11.5 DMAMUX registers

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the DMAMUX base address.

The registers can only be accessed by words (32-bits).

11.5.1 DMAMUX request line multiplexer Channel x Configuration Register DMAMUX register x (DMAMUX_CxCR)

Address offset: 0x04 * x (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[7:0]				
											rw				

Bits 31:5 RESERVED, must be kept at reset value.Reserved, must be kept at reset value.

Bits 4:0 **DMAREQ_ID[4:0]**: DMA REQuest IDentification
Selects the input DMA request. See *Table 29*.

11.5.2 DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

Table 30. DMAMUX register map and reset values

Offset	Register	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
0x000	DMAMUX_C0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x004	DMAMUX_C1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x008	DMAMUX_C2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x00C	DMAMUX_C3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x010	DMAMUX_C4CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x014	DMAMUX_C5CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x018	DMAMUX_C6CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0

Offset	Register	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
0x01C	DMAMUX_C7CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAREQ_ID[4:0]				
	Reset value																												0	0	0	0	0
0x020 - 0x3E8	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

12 Analog to digital converter (ADC)

The STM32WB09xE embeds a 12-bit ADC. The ADC consists in a 12-bit successive approximation analog-to-digital converter (SAR) with 2 x 8 multiplexed channels allowing measurements of up to eight external sources and up to three internal sources.

12.1 Features

- Conversion frequency is up to 1 Msps.
- Three input voltage ranges are supported (0 → 1.2 V, 0 → 2.4 V, 0 → 3.6 V)
- Up to eight analog single ended channels or four analog differential inputs or a mix of both.
- Temperature sensor conversion
- Battery level conversion up to 3.6 V
- Continuous or single acquisition
- ADC Mode conversion only available, programmable in continuous or single mode.
- ADC Down Sampler for multi-purpose applications to improve analog performance while off-loading the CPU (ratio adjustable from 1 to 128).
- A watchdog feature to inform when data is outside thresholds.
- DMA capability.
- Interrupt sources with flags

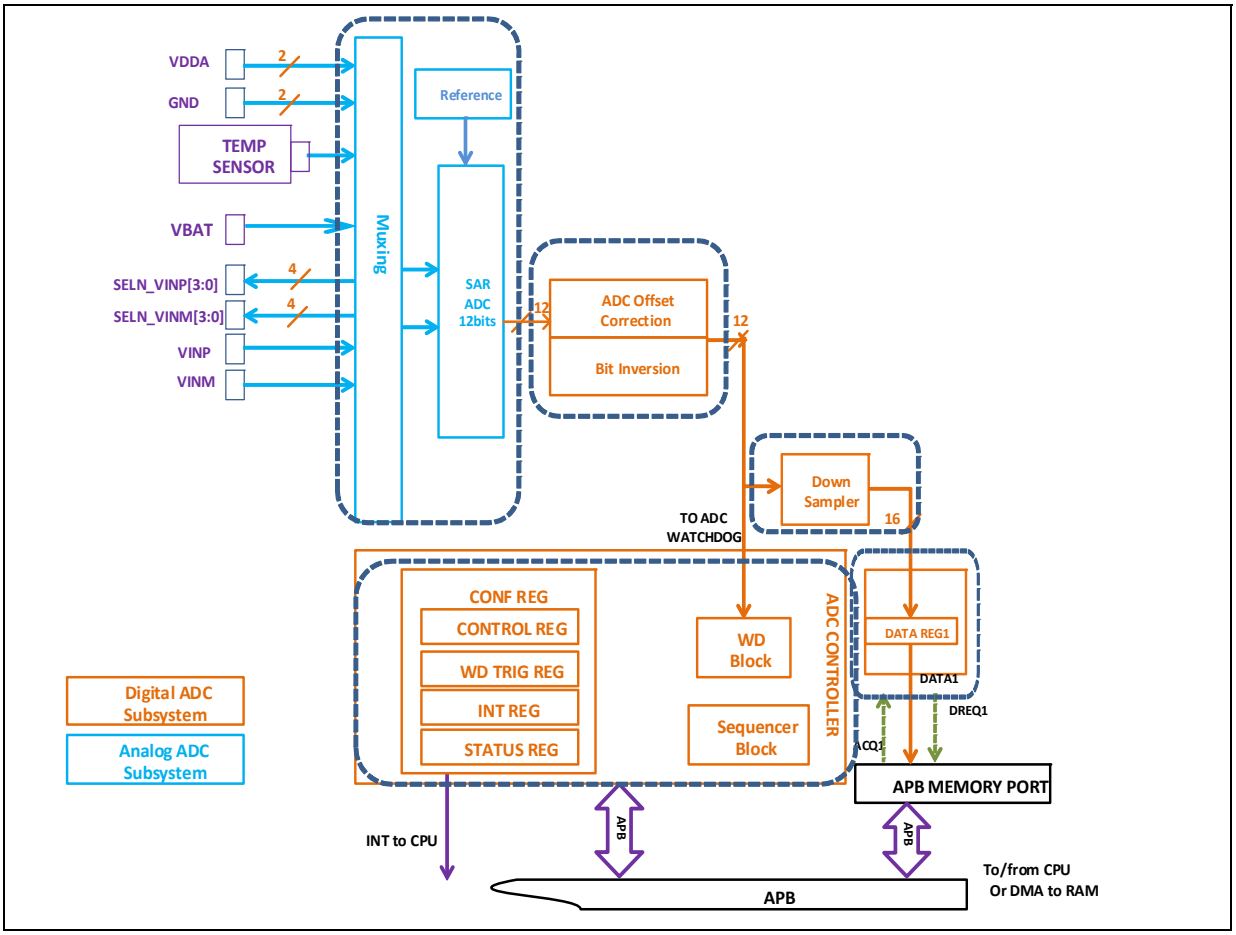
12.2 ADC presentation

Figure 22 shows the top level diagram of the ADC.

The analog ADC is designed to support sixteen possible inputs:

- External signals through ADC_VINPx and ADC_VINMx, where x=0,1,2 or 3
  - Up to 4 differential inputs
  - Up to 8 single-ended inputs one single ended input to interface with a temperature sensor in a range up to 1.2 V
- one single ended input connected to VBAT for battery level detector in a range up to 3.6 V
- two single ended inputs connected to GND for calibration
- two single ended inputs connected to 1.2 V for calibration
- one single ended input connected to Temperature Sensor

Figure 22. ADC top level diagram



The input of the data path can come from the analog ADC through the possible inputs mentioned previously.

The conversion data path can go through a down sampler (for static or low frequency input signals).

Caution: Do not change the configuration registers related to the function in use. Any change done by the user on the different bits are applied immediately, with an immediate effect on the on-going process (conversion, decimator filter or downsampler). This action can lead to unexpected results.

Caution: For VBAT < 2.7 V, the IO Booster needs to be activated to maintain linearity.

12.2.1 Temperature sensor subsystem

The temperature sensor can be used to measure the junction temperature (Tj) of the device. The temperature sensor is internally connected to the ADC input channels, which are used to convert the sensor output voltage to a digital value.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip-to-chip due to process variation. The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. In this way the temperature can be calculated with this formula:

Temperature in Celsius = (Cmeas - C30 + TCK) /10

Where:

TCK is the chuck temperature in 0.1 °C (for example. 30 °C = 300) readable @0x10001E5C.

C30 is the temperature sensor calibration value acquired at 30 °C readable @0x10001E60.

Cmeas is the actual temperature sensor output value converted by ADC.

Note: 1 ADC gain and offset calibration are left as default values, 0xFFF.

2 Refer to Section 27.1: DESIG registers for information about the location where these calibration values are stored.

12.2.2 Battery sensor

The battery sensor can be used to measure the internal battery voltage of the device. The battery input is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value.

The battery sensor range is up to 3.6 V.

The formula for ADC converted data after calibration is the following:

Code = Integer(4096/3.6 * VIN) [clamped at 4095]

As calibration points the VBAT is considered as single negative input with 3.6 V range.

Note: Refer to Section 27.1: DESIG registers for information about the location where these calibration values are stored.

12.2.3 ADC input modes conversion

The ADC is designed to deliver a digital value corresponding to the ratio between the voltage applied on the converted channel and the reference voltage, VDDA. Note that VDDA is also the ADC's power supply. The formula for ADC digital converted data after calibration and offset is the following:

- Single ended input mode

Code = Integer(Slope * VIN) [clamped at 4095]

where Slope for single ended input mode has the following value:

3.6V mode: Slope = 4096/3.6 [calibrated gain = 1/3]

2.4V mode: Slope = 4096/2.4 [calibrated gain = 1/2]

1.2V mode: Slope = 4096/1.25 [calibrated gain =0.96, gain clamped at 1]

- Differential input mode

Code = Integer(Slope * (VINP - VINN)) + 2048 [clamped at 4095]

where Slope for differential input mode has the following value:

3.6V mode: Slope = 2048/3.6 [calibrated gain = 1/3]

2.4V mode: Slope = 2048/2.4 [calibrated gain = 1/2]

1.2V mode: Slope = 2048/1.25 [calibrated gain =0.96, gain clamped at 1]

12.2.4 Steady state input impedance

As the input nature of the ADC is a switched-capacitor, its steady state input impedance is defined as the impedance seen in DC. It depends only on the analog sampling frequency, Fs, and the input capacitor, Cin: Zin = 1/(Cin*Fs).

12.2.5 Input signal sampling transient response

As shown in *Figure 24*, the analog signal path consists of a series resistance (Rext) between source and pin, the internal switch resistor (Rin) and the internal sampling capacitor (Cin). The charging of the capacitor is controlled by Rin. When there is Rext in series, the effective value of charging of Cin is governed by Rin+Rext. So the charging time constant becomes (Rin+Rext)*Cin and the necessary time to reach a given accuracy is longer. The ADC sampling time Tsw is dependent on ADC frequency which is 1/Ts as shown in *Figure 23*.

Figure 23. ADC sampling time Tsw and sampling period Ts

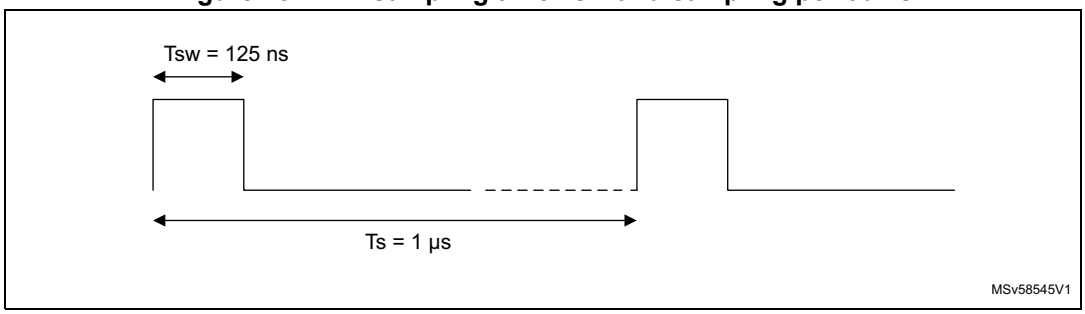
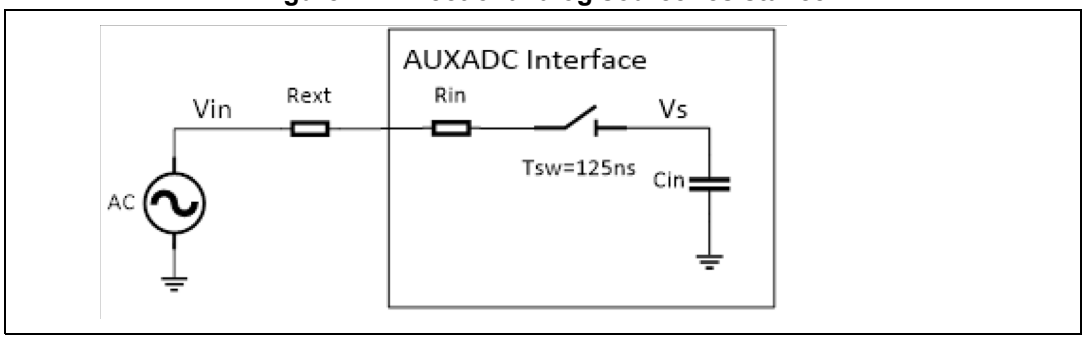


Figure 24. Effect of analog source resistance



Knowing that: Rin=550 Ω, Cin=4 pF, and imposing a maximum sampling error of 1/2 LSB, we can determine the maximum input resistance as below:

$\varepsilon = (Vs-Vin)/Vin = -e^{-t/RC}$

$\ln|\varepsilon| \leq t/(Rext+Rin)*Cin$

$(Rext+Rin)*Cin \leq t/\ln|\varepsilon|$

$(Rext+Rin)*Cin \leq 125e-9/\ln(122e-6)$

$(Rext+Rin)*Cin \leq 14$ ns

$Rext \leq 14ns/4pF - 550\ \Omega$

$Rext \leq 2950\ \Omega$

where: $|\varepsilon| \leq 1/213$

$|\varepsilon| \leq 122e-6$

12.2.6 Calibration points

Calibration values are stored in the system memory for each device by ST during production. Each value consists of a 12-bit unsigned value for the gain and an 8-bit signed value for the offset as follows:

OFFSET[18:12] | GAIN[11:0]

These values can be written inside the registers COMP_x with x=1, 2, 3, 4 to apply a point to a particular ADC input. The COMP_SEL register allows a specific calibration point to be associated to one of the ADC inputs.

The offse tvalue can be written inside the COMP_x register only if it fits in the 7-bit of register field, that means offsetis in [-64, 63]. Otherwise, it can be removed by the output raw data manually as raw_value + offset.

The negative offset values need to be converted as:

offset | 0x80, if the bitfield BIT_INVERT_SN=1(default value)

The table below lists the calibration points and their location in the system memory.

Table 31. Calibration points

Calibration point	Address location
VINPx - VINMx range 1.2 V	0x10001E00
VINMx range 1.2 V	0x10001E04
VINPx range 1.2 V	0x10001E08
VINPx - VINMx range 2.4 V	0x10001E0C
VINMx range 2.4 V	0x10001E10
VINPx range 2.4 V	0x10001E14
VINPx - VINMx range 3.6 V	0x10001E18
VINMx range 3.6 V	0x10001E1C
VINPx range 3.6 V	0x10001E20

Note: 1 Previous version of the calibration points has the offset in 7-bit signed. This version can be recognized as the user can read at address 0x10001EFC the value 0.

2 Refer to Section 27.1: DESIG registers for information about the location where these calibration values are stored.

12.2.7 Down sampler (DS)

This Down Sampler is a simple averaging filter which can divide the ADC frequency by 1 to 128 by power of 2.

The goal is to handle multiple ADC samples and average then into a single data with increased data width ranging from 12-bit to 16-bit.

The Down Sampler increases the data precision but reduces the output data rate.

Note: A constraint on the ratio between APB system clock (F_{PCLK}) and the output data rate (DRout) must be respected:

- In ADC mode, the ratio to respect is F_{PCLK} / DRout ≥ 4.

Example: F_{PCLK} must be at least 2 MHz to have a DRout = 500 kHz.

If the DMA is not used to get the data output by the down sampler path, the CPU needs to be clocked at a frequency ratio high enough (taking into account bus matrix latency) to avoid missing samples.

12.3 Interrupts

There are 5 maskable interrupts generated by the ADC block. These interrupts are combined to produce one single interrupt output which is the only interrupt line from the ADC to the CPU.

Table 32. ADC interrupt requests

Interrupt event	Event flag	Interrupt / flag clearing method	Interrupt enable control bit
ADC end of Conversion (Test mode only)	EOC_IRQ	Write 1 on EOC_IRQ bit	EOC_IRQ_ENA
Down Sampler end of conversion	EODS_IRQ	Write 1 on EODS_IRQ bit	EODS_IRQ_ENA
End of conversion sequence	EOS_IRQ	Write 1 on EOS_IRQ bit	EOS_IRQ_ENA
Analog Watchdog event	AWD_IRQ	Write 1 on AWD_IRQ bit	AWD_IRQ_ENA
Down Sampler overrun	OVR_DS_IRQ	Write 1 on OVR_DS_IRQ bit	OVR_DS_IRQ_ENA

12.4 DMA interface

The ADC has one DMA channel interface to get Down Sampler data output value.

The DMA feature is enabled by software through CONF register by DMA_DS_ENA bit.

When DMA feature is disabled, the data can be read by the CPU through the corresponding APB register.

12.5 ADC mode

ADC mode is the only conversion mode available. As a consequence no concurrent mode is also available.

The input signal can come from:

- 8 single external channels (4 positive + 4 negative, or 4 differential when coupled).
- VBAT: negative single ended input to 3.6 V
- Temperature Sensor: positive single ended input to 1.2 V

The conversion can be continuous or single mode.

12.5.1 ADC mode overview

Presentation

The ADC mode has the following characteristics:

- The input in the ADC mode can be the eight external channels and the two internal sources (VBAT and Temperature sensor).
- The data path goes from the ADC to the down sampler.
- The converted data is output in the DS_DATAOUT register.
- The output data rates are in the range 117 sps to 1 Msps.
- The 12-bit converted data can be extended up to 16-bit data thanks to the down sampler. However, in this case, the output data rate is decreased.
- A regular sequence of conversion can be executed in single of continuous mode.
  - A regular sequence consists in chaining ADC conversions on any ADC input channel and in any order.
  - A regular sequence can chain up to 16 conversions.
  - The source of the input for each conversion of the sequence is selected through SEQx bit field in SEQ_1 and SEQ_2 registers.
  - This regular sequence can be run once or repeated continuously by setting the CONT bit in CONF register.

ADC mode usage

This paragraph describes the process to use the ADC mode:

- Power on the ADC if not yet done by setting the ADC_ON_OFF bit in the CTRL register.
- Program the targeted data rate through SAMPLE_RATE and DS_CONF registers.
- Program the input voltage selections through SWITCH register.
- Program the COMP_1 to COMP_4 and the COMP_SEL registers.
- Program the ADC mode through the OP_MODE bit field in the CONF register.
- Program the targeted regular sequence (up to 16 chained conversions) through SEQ_1 and SEQ_2 registers.
- Specify the length of the sequence in SEQ_LEN bit field in CONF (from 0 for one conversion to 0xF for sixteen conversions).

Note: To have more than one conversion, ensure the bit SEQUENCE is well at 1 in CONF register.

- Program the CONT bit and the SEQ_LEN bit field in the CONF register, considering SEQUENCE bit is always set) depending on the wished sequence:
  - CONT = 0 and SEQ_LEN = 0 to have a single conversion on a single channel.
  - CONT = 0 and SEQ_LEN > 0 to have a single run of a sequence chaining several conversions on different channels/sources.
  - CONT = 1 and SEQ_LEN = 0 to have a continuous conversion of a single channel/source.
  - CONT = 1 and SEQ_LEN > 0 to have a continuous run of sequence chaining several conversions on different channels/sources.
- Launch the programmed regular sequence by setting the START_CONV bit in CTRL register.
- Each time a data is available at the output of the Sown Sampler, the data is stored in the DS_DATAOUT register and the EODS flag is set (as analog mode goes through the Down Sampler).
- To get the converted values:
  - Either the DMA is enabled on DS data path (through DMA_DS_ENA bit in CONF register) and DMA copies the converted data in RAM at the end of each data conversion
  - Or the software has enabled the EODS_IRQ interrupt and is able to get the data from DS_DATAOUT register before a new converted data is generated.

Note: If the CPU does not manage to get the converted data before a new converted data is generated, the OVR_DS_IRQ flag is raised to inform a data has been lost.

The software can program the hardware behavior in case of overrun through the OVR_DS_CFG bit in CONF register:

if 0, the previous data is kept, the new one is lost.

if 1, the previous data is lost, the new one is kept.

- Each time the regular sequence is completed, the EOS_IRQ flag is raised (and may generate an interrupt if enabled).
- If the sequence is a single sequence (CONT=0), the ADC stops at the end of the sequence and does not restart until START_CONV bit is set again.
- If continuous conversion is enabled (CONT = 1), the ADC restarts a new sequence, and data conversion continues until the software stops it by setting the

STOP_OP_MODE bit in the CTRL register. In this case, the ADC stops immediately, and the data from any ongoing conversion is discarded.

12.6 ADC registers

12.6.1 Version register (VERSION_ID)

Address offset: 0x00

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VERSION_ID[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **VERSION_ID[7:0]**: version of the embedded IP.

12.6.2 ADC configuration register (CONF)

Address offset: 0x04

Reset value: 0x000A 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAMPLE_RATE_MSB			Res.	ADC_CONT_1V2	BIT_INVERT_DIFF	BIT_INVERT_SN	Res.
								rw	rw	rw		rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVR_DS_CFG	Res.	DMA_DS_ENA	SAMPLE_RATE[1:0]		Res.	Res.	Res.	Res.	SMPS_SYNCHRO_ENA	SEQ_LEN[3:0]				SEQUENCE	CONT
rw		rw	rw						rw	rw	rw	rw	rw	rw	rw

Bits 31: 24 Reserved

Bits 23:21 **SAMPLE_RATE_MSB**: sample rate most significant bit
This field is an extension of SAMPLE_RATE definition in bits 12,11 of CONF register. It impacts the conversion rate of ADC (F_ADC). See SAMPLE_RATE bits for the full description

Bit 20 Reserved, must be kept at reset value.

Bit 19 **ADC_CONT_1V2**: select the input sampling method:
0: Sampling time is 125 ns regardless of the sampling period
1: Sampling time is a function of the sampling period (default)

Bit 18 **BIT_INVERT_DIFF**: invert bit to bit the ADC data output (1's complement) when a differential input is connected to the ADC:
0: no inversion (default)
1: enable the inversion

Bit 17 **BIT_INVERT_SN**: invert bit to bit the ADC data output (1's complement) when a single negative input is connected to the ADC:
0: no inversion
1: enable the inversion (default)

Bit 16 Reserved, must be kept at reset value.

Bit 15 **OVR_DS_CFG**: Down Sampler overrun configuration:
0: the previous data is kept, the new one is lost
1: the previous data is lost, the new one is kept

Bit 14 Reserved, must be kept at reset value.

Bit 13 **DMA_DS_EN**: enable the DMA mode for the Down Sampler data path:
0: DMA mode is disabled
1: DMA mode is enabled

Bits 12:11 **SAMPLE_RATE[1:0]**: conversion rate of ADC (F_ADC):
F_ADC = F_ADC_CLK/(16 + 16*SAMPLE_RATE_MSB + 4*SAMPLE_RATE)
where F_ADC_CLK is the analog ADC clock frequency. By default F_ADC_CLK is 16 MHz frequency.

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **SMPS_SYNCHRO_ENA**: synchronize the ADC start conversion with a pulse generated by the SMPS:
0: SMPS synchronization is disabled for all ADC clock frequencies
1: SMPS synchronization is enabled
Note: SMPS_SYNCHRO_ENA must be 0 when PWRC_CR5.NOSMPS = 1.

Bits 5:2 **SEQ_LEN[3:0]**: number of conversions in a regular sequence:
0000: 1 conversion, starting from SEQ0
0001: 2 conversions, starting from SEQ0
...
1111: 16 conversions, starting from SEQ0

Bit 1 **SEQUENCE**: enable the sequence mode (active by default):
0: sequence mode is disabled, only SEQ0 is selected
1: sequence mode is enabled, conversions from SEQ0 to SEQx with x=SEQ_LEN
Note: clearing this bit is equivalent to SEQUENCE=1 and SEQ_LEN=0000. Ideally, this bit can be kept high as redundant with keeping high and setting SEQ_LEN=0000.

Bit 0 **CONT**: regular sequence runs continuously when ADC mode is enabled:
0: enable the single conversion: when the sequence is over, the conversion stops
1: enable the continuous conversion: when the sequence is over, the sequence starts again until the software sets the CTRL.STOP_OP_MODE bit.

12.6.3 ADC control register (CTRL)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP_OP_MODE	START_CONV	ADC_ON_OFF
													t	t	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **STOP_OP_MODE** (1): stop the on-going OP_MODE (ADC mode):
0: no effect
1: stop on-going ADC mode
Note: this bit is set by software and cleared by hardware.

Bit 1 **START_CONV** (1): generate a start pulse to initiate an ADC conversion:
0: no effect
1: start the ADC conversion
Note: this bit is set by software and cleared by hardware.

Bit 0 **ADC_ON_OFF**:
0: power off the ADC
1: power on the ADC

1. When setting the STOP_MODE_OP, the user has to wait around 10 us before to start a new ADC conversion by setting the START_CONV bit.

12.6.4 ADC input voltage switch selection register (SWITCH)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SE_VIN_7[1:0]		SE_VIN_6[1:0]		SE_VIN_5[1:0]		SE_VIN_4[1:0]		SE_VIN_3[1:0]		SE_VIN_2[1:0]		SE_VIN_1[1:0]		SE_VIN_0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **SE_VIN_7[1:0]**: input voltage for VINP[3]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 13:12 **SE_VIN_6[1:0]**: input voltage for VINP[2]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 11:10 **SE_VIN_5[1:0]**: input voltage for VINP[1]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 9:8 **SE_VIN_4[1:0]**: input voltage for VINP[0]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 7:6 **SE_VIN_3[1:0]**: input voltage for VINM[3] / VINP[3]-VINM[3]
- 00: Vinput = 1.2 V
- 01: reserved (not used for this cut)
- 10: Vinput = 2.4 V
- 11: Vinput = 3.6 V

Bits 5:4 **SE_VIN_2[1:0]**: input voltage for VINM[2] / VINP[2]-VINM[2]
00: Vinput = 1.2 V
01: reserved (not used for this cut)
10: Vinput = 2.4 V
11: Vinput = 3.6 V

Bits 3:2 **SE_VIN_1[1:0]**: input voltage for VINM[1] / VINP[1]-VINM[1]
00: Vinput = 1.2 V
01: reserved (not used for this cut)
10: Vinput = 2.4 V
11: Vinput = 3.6 V

Bits 1:0 **SE_VIN_0[1:0]**: input voltage for VINM[0] / VINP[0]-VINM[0]
00: Vinput = 1.2 V
01: reserved (not used for this cut)
10: Vinput = 2.4 V
11: Vinput = 3.6 V

12.6.5 Down sampler configuration register (DS_CONF)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS_WIDTH[2:0]			DS_RATIO[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:3 **DS_WIDTH[2:0]**: program the Down Sampler width of data output (DSDTATA)
- 000: DS_DATA output on 12-bit (default)
- 001: DS_DATA output on 13-bit
- 010: DS_DATA output on 14-bit
- 011: DS_DATA output on 15-bit
- 100: DS_DATA output on 16-bit
- 1xx: reserved

Bits 2:0 **DS_RATIO[2:0]**: program the Down Sampler ratio (N factor)
- 000: ratio = 1, no down sampling (default)
- 001: ratio = 2
- 010: ratio = 4
- 011: ratio = 8
- 100: ratio = 16
- 101: ratio = 32
- 110: ratio = 64
- 111: ratio = 128

12.6.6 ADC sequence programming 1 register (SEQ_1)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEQ7[3:0]				SEQ6[3:0]				SEQ5[3:0]				SEQ4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ3[3:0]				SEQ2[3:0]				SEQ1[3:0]				SEQ0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SEQ7[3:0]**: channel number code for 8th conversion of the sequence.
See SEQ0 for code detail.

Bits 27:24 **SEQ6[3:0]**: channel number code for 7th conversion of the sequence.
See SEQ0 for code detail.

Bits 23:20 **SEQ5[3:0]**: channel number code for 6th conversion of the sequence.
See SEQ0 for code detail.

Bits 19:16 **SEQ4[3:0]**: channel number code for 5th conversion of the sequence.
See SEQ0 for code detail.

Bits 15:12 **SEQ3[3:0]**: channel number code for 4th conversion of the sequence.
See SEQ0 for code detail.

Bits 11:8 **SEQ2[3:0]**: channel number code for 3rd conversion of the sequence.
See SEQ0 for code detail.

Bits 7:4 **SEQ1[3:0]**: channel number code for second conversion of the sequence.
See SEQ0 for code detail.

Bits 3:0 **SEQ0[3:0]**: channel number code for first conversion of the sequence
0000: VINM[0] to ADC single negative input
0001: VINM[1] to ADC single negative input
0010: VINM[2] to ADC single negative input
0011: VINM[3] to ADC single negative input
0100: VINP[0] to ADC single positive input
0101: VINP[1] to ADC single positive input
0110: VINP[2] to ADC single positive input
0111: VINP[3] to ADC single positive input
1000: VINP[0]-VINM[0] to ADC differential input
1001: VINP[1]-VINM[1] to ADC differential input
1010: VINP[2]-VINM[2] to ADC differential input
1011: VINP[3]-VINM[3] to ADC differential input
1100: VBAT - Battery level detector
1101: Temperature sensor
111x: reserved

12.6.7 ADC sequence programming 2 register (SEQ_2)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEQ15[3:0]				SEQ14[3:0]				SEQ13[3:0]				SEQ12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQ11[3:0]				SEQ10[3:0]				SEQ9[3:0]				SEQ8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SEQ15[3:0]**: channel number code for 16th conversion of the sequence.
See SEQ8 for code detail.

Bits 27:24 **SEQ14[3:0]**: channel number code for 15th conversion of the sequence.
See SEQ8 for code detail.

Bits 23:20 **SEQ13[3:0]**: channel number code for 14th conversion of the sequence.
See SEQ8 for code detail.

Bits 19:16 **SEQ12[3:0]**: channel number code for 13th conversion of the sequence.
See SEQ8 for code detail.

Bits 15:12 **SEQ11[3:0]**: channel number code for 12th conversion of the sequence.
See SEQ8 for code detail.

Bits 11:8 **SEQ10[3:0]**: channel number code for 11th conversion of the sequence.
See SEQ8 for code detail.

Bits 7:4 **SEQ9[3:0]**: channel number code for 10th conversion of the sequence.
See SEQ8 for code detail.

Bits 3:0 **SEQ8[3:0]**: channel number code for 9th conversion of the sequence
0000: VINM[0] to ADC single negative input
0001: VINM[1] to ADC single negative input
0010: VINM[2] to ADC single negative input
0011: VINM[3] to ADC single negative input
0100: VINP[0] to ADC single positive input
0101: VINP[1] to ADC single positive input
0110: VINP[2] to ADC single positive input
0111: VINP[3] to ADC single positive input
1000: VINP[0]-VINM[0] to ADC differential input
1001: VINP[1]-VINM[1] to ADC differential input
1010: VINP[2]-VINM[2] to ADC differential input
1011: VINP[3]-VINM[3] to ADC differential input
1100: VBAT - Battery level detector
1101: temperature sensor
111x: reserved

12.6.8 ADC gain and offset correction 1 register (COMP_1)

Address offset: 0x28

Reset value: 0x0000 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSFFSET1[7:4]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSFFSET1[3:0]				GAIN1[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET1[7:0]**: first calibration point: signed offset compensation[6:0]

Bits 11:0 **GAIN1[11:0]**: first calibration point: gain AUXADC_GAIN_1V2[11:0]

Note: Refer to Section 27.1: DESIG registers for information about the location where the calibration values are stored.

12.6.9 ADC gain and offset correction 2 register (COMP_2)

Address offset: 0x2C

Reset value: 0x0000 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSFFSET2[7:4]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSFFSET2[3:0]				GAIN2[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET2[7:0]**: first calibration point: signed offset compensation[6:0]

Bits 11:0 **GAIN2[11:0]**: second calibration point: gain AUXADC_GAIN_1V2[11:0]

Note: Refer to Section 27.1: DESIG registers for information about the location where the calibration values are stored.

12.6.10 ADC gain and offset correction 3 register (COMP_3)

Address offset: 0x30

Reset value: 0x0000 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSFFSET3[7:4]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSFFSET3[3:0]				GAIN3[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET3[7:0]**: first calibration point: signed offset compensation[6:0]

Bits 11:0 **GAIN3[11:0]**: third calibration point: gain AUXADC_GAIN_1V2[11:0]

Note: Refer to Section 27.1: DESIG registers for information about the location where the calibration values are stored.

12.6.11 ADC gain and offset correction 4 register (COMP_4)

Address offset: 0x34

Reset value: 0x0000 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSFFSET4[7:4]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSFFSET4[3:0]				GAIN4[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET4[7:0]**: first calibration point: signed offset compensation[6:0]

Bits 11:0 **GAIN4[11:0]**: fourth calibration point: gain AUXADC_GAIN_1V2[11:0]

Note: Refer to Section 27.1: DESIG registers for information about the location where the calibration values are stored.

12.6.12 ADC gain and offset selection register (COMP_SEL)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET_GAIN8[1:0]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET_GAIN7[1:0]		OFFSET_GAIN6[1:0]		OFFSET_GAIN5[1:0]		OFFSET_GAIN4[1:0]		OFFSET_GAIN3[1:0]		OFFSET_GAIN2[1:0]		OFFSET_GAIN1[1:0]		OFFSET_GAIN0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **OFFSET_GAIN8[1:0]**: gain / offset used in ADC differential mode with Vinput range = 3.6V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 15:14 **OFFSET_GAIN7[1:0]**: gain / offset used in ADC single positive mode with Vinput range = 3.6V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 13:12 **OFFSET_GAIN6[1:0]**: gain / offset used in ADC single negative mode with Vinput range = 3.6V. This field also selects the gain/offset for VBAT input:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 11:10 **OFFSET_GAIN5[1:0]**: gain / offset used in ADC differential mode with Vinput range = 2.4V:
- 00: OFFSET1 and GAIN1 from COMP_1
- 01: OFFSET2 and GAIN2 from COMP_2
- 10: OFFSET3 and GAIN3 from COMP_3
- 11: OFFSET4 and GAIN4 from COMP_4

Bits 9:8 **OFFSET_GAIN4[1:0]**: gain / offset used in ADC single positive mode with Vinput range = 2.4V:
00: OFFSET1 and GAIN1 from COMP_1
01: OFFSET2 and GAIN2 from COMP_2
10: OFFSET3 and GAIN3 from COMP_3
11: OFFSET4 and GAIN4 from COMP_4

Bits 7:6 **OFFSET_GAIN3[1:0]**: gain / offset used in ADC single negative mode with Vinput range = 2.4V:
00: OFFSET1 and GAIN1 from COMP_1
01: OFFSET2 and GAIN2 from COMP_2
10: OFFSET3 and GAIN3 from COMP_3
11: OFFSET4 and GAIN4 from COMP_4

Bits 5:4 **OFFSET_GAIN2[1:0]**: gain / offset used in ADC differential mode with Vinput range = 1.2V:
00: OFFSET1 and GAIN1 from COMP_1
01: OFFSET2 and GAIN2 from COMP_2
10: OFFSET3 and GAIN3 from COMP_3
11: OFFSET4 and GAIN4 from COMP_4

Bits 3:2 **OFFSET_GAIN1[1:0]**: gain / offset used in ADC single positive mode with Vinput range = 1.2V. This field also selects the gain/offset for Temperature Sensor input:
00: OFFSET1 and GAIN1 from COMP_1
01: OFFSET2 and GAIN2 from COMP_2
10: OFFSET3 and GAIN3 from COMP_3
11: OFFSET4 and GAIN4 from COMP_4

Bits 1:0 **OFFSET_GAIN0[1:0]**: gain / offset used in ADC single negative mode with Vinput range = 1.2V:
00: OFFSET1 and GAIN1 from COMP_1
01: OFFSET2 and GAIN2 from COMP_2
10: OFFSET3 and GAIN3 from COMP_3
11: OFFSET4 and GAIN4 from COMP_4

12.6.13 ADC watchdog threshold register (WD_TH)

Address offset: 0x3C

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WD_HT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WD_LT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WD_HT[11:0]**: analog watchdog high level threshold.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **WD_LT**[11:0]: analog watchdog low level threshold.

12.6.14 ADC watchdog configuration register (WD_CONF)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD_CHX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **AWD_CHX[15:0]**: analog watchdog channel selection to define which input channel(s) need to be guarded by the watchdog.

Bit0: VINM[0] to ADC negative input
Bit1: VINM[1] to ADC negative input
Bit2: VINM[2] to ADC negative input
Bit3: VINM[3] to ADC negative input
Bit4: GND to ADC negative input
Bit5: VBAT to ADC negative input
Bit6: GND to ADC negative input
Bit7: VDDA to ADC negative input
Bit8: VINP[0] to ADC positive input
Bit9: VINP[1] to ADC positive input
Bit10: VINP[2] to ADC positive input
Bit11: VINP[3] to ADC positive input
Bit12: VBAT to ADC positive input
Bit13: TEMP to ADC positive input
Bit14: GND to ADC positive input
Bit15: VDDA to ADC positive input

12.6.15 Down sampler data out register (DS_DATAOUT)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS_DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DS_DATA[15:0]**: contain the converted data at the output of the Down Sampler.

12.6.16 ADC interrupt status register (IRQ_STATUS)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_DS_IRQ	AWD_IRQ	EOS_IRQ	Res.	EODS_IRQ	EOC_IRQ
										rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **OVR_DS_IRQ**: set to indicate a Down Sampler overrun (at least one data is lost)
When read, provide the status of the interrupt:
0: no overrun occurred
1: overrun occurred
Writing this bit clears the status of the interrupt:
0: no effect
1: clear the interrupt

Bit 4 **AWD_IRQ**: set when an analog watchdog event occurs.
When read, provide the status of the interrupt:
0: no analog watchdog event occurred
1: analog watchdog event has occurred
Writing this bit clears the status of the interrupt:
0: no effect
1: clear the interrupt

Bit 3 **EOS_IRQ**: set when a sequence of conversion is completed.
When read, provide the status of the interrupt:
0: sequence of conversion is not completed
1: sequence of conversion is completed
Writing this bit clears the status of the interrupt:
0: no effect
1: clear the interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EODS_IRQ**: set when the Down Sampler conversion is completed.
When read, provide the status of the interrupt:
0: Down Sampler conversion is not completed
1: Down Sampler conversion is completed
Writing this bit clears the status of the interrupt:
0: no effect
1: clear the interrupt

Bit 0 **EOC_IRQ** (Used in test mode only): set when the ADC conversion is completed.
When read, provide the status of the interrupt:
0: ADC conversion is not completed
1: ADC conversion is completed
Writing this bit clears the status of the interrupt:
0: no effect
1: clear the interrupt

12.6.17 ADC Interrupt enable register (IRQ_ENABLE)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_DS_IRQ_ENA	AWD_IRQ_ENA	EOS_IRQ_ENA	Res.	EODS_IRQ_ENA	EOC_IRQ_ENA
										rw	rw	rw		rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **OVR_DS_IRQ_ENA**: Down Sampler overrun interrupt enable:
- 0: Down Sampler interrupt is disabled
- 1: Down Sampler interrupt is enabled

Bit 4 **AWD_IRQ_ENA**: analog watchdog interrupt enable:
- 0: analog watchdog interrupt is disabled
- 1: analog watchdog interrupt is enabled

Bit 3 **EOS_IRQ_ENA**: End of regular sequence interrupt enable:
- 0: EOS interrupt is disabled
- 1: EOS interrupt is enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EODS_IRQ_ENA**: End of conversion interrupt enable for the Down Sampler output:
- 0: EODF interrupt is disabled
- 1: EODF interrupt is enabled

Bit 0 **EOC_IRQ_ENA** (Used in test mode only): End of ADC conversion interrupt enable:
- 0: EOC interrupt is disabled
- 1: EOC interrupt is enabled

12.6.18 ADC register map

Refer to *Table 3: Memory map and peripheral register boundary addresses* for the ADC base address location in the STM32WB09xE.

Table 33. ADC register map and reset values

Offset	Register	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
0x00	VERSION_ID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VERSION_ID[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0x04	CONF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAMPLE_RATE_MSB			Res.	ADC_CONT_1V2	BIT_INVERT_DIFF	BIT_INVERT_SN	Res.	OVR_DS_CFG	Res.	DMA_DS_ENA	SAMPLE_RATE[1:0]		Res.	Res.	Res.	SMPS_SYNCHRO_ENA	SEQ_LEN[3:0]				SEQUENCE	CONT.
	Reset value									0	0	0		1	0	1		0		0	0	0					0	0	0	0	0	1	0
0x08	CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP_OP_MODE	START_CONV	ADC_ON_OFF
	Reset value																														0	0	0
0x14	SWITCH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SE_VIN_7[1:0]		SE_VIN_6[1:0]		SE_VIN_5[1:0]		SE_VIN_4[1:0]		SE_VIN_3[1:0]		SE_VIN_2[1:0]		SE_VIN_1[1:0]		SE_VIN_0[1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DS_CONF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS_WIDTH[2:0]			DS_RATIO[2:0]		
	Reset value																											0	0	0	0	0	0
0x20	SEQ_1	SEQ7[3:0]				SEQ6[3:0]				SEQ5[3:0]				SEQ4[3:0]				SEQ3[3:0]				SEQ2[3:0]				SEQ1[3:0]				SEQ0[3:0]			
	Reset value	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
0x24	SEQ_2	SEQ15[3:0]				SEQ14[3:0]				SEQ13[3:0]				SEQ12[3:0]				SEQ11[3:0]				SEQ10[3:0]				SEQ9[3:0]				SE80[3:0]			
	Reset value	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
0x28	COMP_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET1[:0]								GAIN1[11:0]											
	Reset value													0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1

Table 33. ADC register map and reset values (continued)

Offset	Register	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
0x2C	COMP_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET2[:0]								GAIN2[11:0]											
	Reset value													0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1
0x30	COMP_3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET3[:0]								GAIN3[11:0]											
	Reset value													0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1
0x34	COMP_4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET4[:0]								GAIN4[11:0]											
	Reset value													0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1
0x38	COMP_SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET_GAIN8[1:0]		OFFSET_GAIN7[1:0]		OFFSET_GAIN6[1:0]		OFFSET_GAIN5[1:0]		OFFSET_GAIN4[1:0]		OFFSET_GAIN3[1:0]		OFFSET_GAIN2[1:0]		OFFSET_GAIN01[1:0]		OFFSET_GAIN0[1:0]	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	WD_TH	Res.	Res.	Res.	Res.	WD_HT[11:0]												Res.	Res.	Res.	Res.	WD_LT[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x40	WD_CONF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD_CHX[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	DS_DATAOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS_DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_DS_IRQ	AWD_IRQ	EOS_IRQ	Res.	EODS_IRQ	EOC_IRQ
	Reset value																											0	0	0		0	0
0x50	IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_DS_IRQ_EN	AWD_IRQ_ENA	EOS_IRQ_ENA	Res.	EODS_IRQ_ENA	EOC_IRQ_ENA
	Reset value																											0	0	0		0	0

13 Public key accelerator (PKA)

13.1 Introduction

The public key accelerator (PKA) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (Elliptic Curve Cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain. All needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

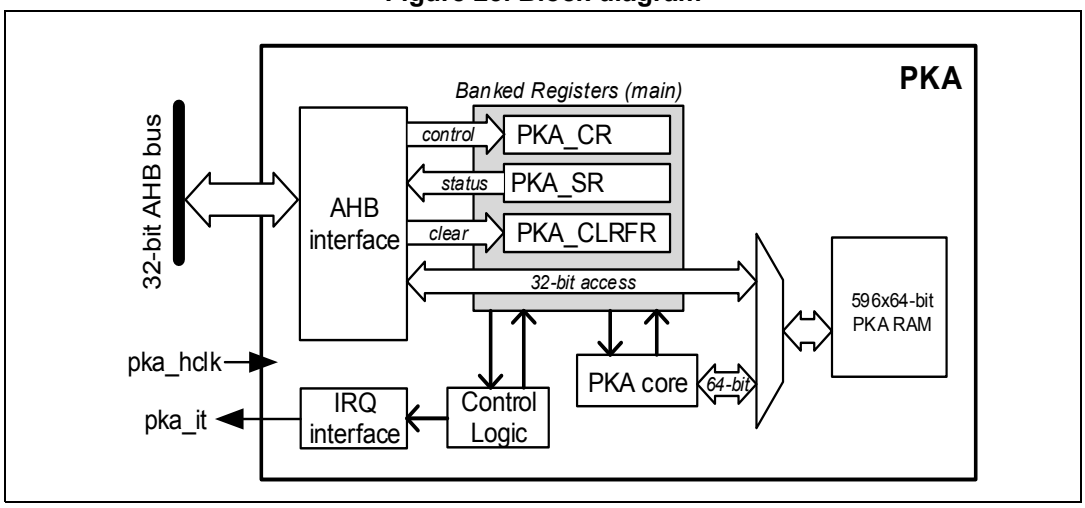
13.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA Chinese Remainder Theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC.
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication.
- Built-in Montgomery domain inward and outward transformations.
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise, for writes, an AHB bus error is generated, and write accesses are ignored).

13.3 PKA functional description

Figure 25 shows the block diagram of the public key accelerator.

Figure 25. Block diagram



PKA can be used for accelerating Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) and Elliptic Curve Cryptography (ECC) operations over a GF(p) field for operand sizes up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

A memory of 38144 bytes (596 words of 64 bits) is used for providing initial data to the PKA, and for holding the results after computation is completed. We call this memory the PKA RAM. Access is done though the PKA AHB interface.

A typical computing sequence by the PKA is composed by the following steps:

1. Calculation of R2 (mod n) parameter to enter the Montgomery domain.
2. Conversion of inputs into the Montgomery residue-system representation.
3. Computation of the requested primitive (modular exponentiation for RSA/DH and point multiplication for ECC).
4. Conversion out from Montgomery representation.

If several operations are computed sequentially using the same modulus n, the first step can be skipped, since the PKA can store the last computed R2 (mod n). User software is also able to load precomputed values of R2 (mod n) directly into the PKA.

13.3.1 Enabling/disabling PKA

Setting the EN bit (PKA_CR[0]) to ‘1’ enables the PKA peripheral. When EN=’0’, the PKA peripheral is reset, no operation can be started and the PKA memory cannot be accessed.

Clearing EN bit while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA RAM (described here below) is not guaranteed.

13.3.2 PKA RAM

The PKA RAM is an internal memory block of 38144 bytes which is organized as 596 words of 64 bits. Each 64-bit word has an address associated with it, from decimal offset 0 to 595.

The offset address to access the first word of the memory block is 0x400. External masters may access this internal memory block from the PKA AHB interface.

Note: Only 32 bits access are supported.

Accesses are allowed only while there is no computation in progress (BUSY = 0). During computation the internal memory block is accessed by the PKA core itself.

13.3.3 Executing a PKA operation

PKA is able to perform 21 types of operations. Each of those PKA operation is executed using the following procedure:

1. load initial data into the PKA internal RAM.
2. load the MODE[5:0] field specifying the operation which is to be executed and assert the START bit. Both fields are in the PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set, which indicates that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

The operations and their corresponding input data and results are described in *Section 13.4: Operating modes*. When selecting an illegal or unknown operation in step *2,*

the step *3* (PROCENDF=1) never happens and BUSY is reset automatically after few clock cycles. See *Section 13.3.5: PKA error management* for details.

13.3.4 Security level

The operations for which a constant time version is available (SECLVL=1) (see *Table 34: Operating modes*) are executed in constant time. For MODE[5:0] = 000000 or MODE[5:0] = 000010 operations, to make the modular exponentiation operations constant time, the exponent size must be set equal to the size of the modulus (see *Section 13.4.2: Compute modular exponentiation*). For MODE[5:0] = 100000 or MODE[5:0] = 100010 operations, to make the ECC scalar multiplication operations constant time, the scalar size must be set equal to the size of the order (see *Section 13.4.3: Compute the ECC scalar multiplication*).

13.3.5 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range or access to PKA registers fall outside the expected range. In this case the Address Error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM Error Flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return zero, while writes are ignored.
- During following sensitive operations: Arithmetic Comparison, ECDSA Signature Verification2, ECC Point Check, ECDSA Signature Verification if a fault occurs, then an error code is written in RAM. When the error code at the end of the operation does not correspond to any of the possible expected values, then the fault check procedure has failed. This scenario should be treated as an attempt to skip the fault check procedure and thus as a fault itself. FAULTFSMF is set indicating that a fault compromised the integrity of the registers driving the operation execution. FAULTERRORCODEF is set indicating that a fault altered the normal execution flow of the operation and that the PKA did not write the final error code.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see *Section 13.5.1: PKA interrupts* for details). ADDRERRF, RAMERRF, errors are cleared by setting the corresponding bit in PKA_CLRFR, while FAULTFSMF and FAULTERRORCODEF are cleared by resetting EN bit in PKA_CR register.

The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

13.4 Operating modes

There are 21 types of operations which the PKA can perform. Each of these operating modes has an associated code which is to be written to the MODE[5:0] field in the PKA_CR register.

Table 34. Operating modes

MODE[5:4]	MODE[3:0]	Operation performed	SECLVL=1
00	0000	Compute Montgomery parameter and modular exponentiation	yes
00	0001	Compute Montgomery parameter	no
00	0010	Compute modular exponentiation only (Montgomery parameter should be loaded)	yes
10	0000	Compute Montgomery parameter and compute ECC kP operation	yes
10	0010	Compute the ECC kP primitive only (Montgomery parameter should be loaded)	yes
10	0011	ECC complete addition	yes
10	0100	ECDSA signature	yes
10	0101	ECDSA signature verification	no
10	0111	ECC double base ladder	no
10	1000	Point Check	no
00	0111	RSA CRT exponentiation	no
00	1000	Modular inversion	no
00	1001	Arithmetic addition	no
00	1010	Arithmetic subtraction	no
00	1011	Arithmetic multiplication	no
00	1100	Comparison	no
00	1101	Modular reduction	no
00	1110	Modular addition	yes
00	1111	Modular subtraction	yes
01	0000	Montgomery Multiplication	yes

The format of the input data and the results in the PKA RAM are specified for each operating mode in the following sections.

The size of the operands is configurable. For RSA operations (where MODE[5]=0), the maximum “rsa_size” is 4160 bits. For ECC operations (where MODE[5]=1), the maximum “ecc_size” is 640.

In the tables below, the acronyms ROS (RSA Operand Size) and EOS (ECC Operand Size) indicate how many words the given field includes.

ROS = (rsa_size / 64) + 1

EOS = (ecc_size / 64) + 1

Fractional results are rounded up to the nearest integer since the PKA's processing is based on 64-bit words. The maximum ROS is 66 (4160-bit max exponent size), while the maximum EOS is 11 (640-bit max operand size).

For example, if you want to compute RSA with an operand of 1024 bits then ROS equals to 17. If you want to compute ECC with an operand of 192 bits, then EOS = 4.

Note: For the input fields whose size is ROS, ROS/2, or EOS, an additional word which is entirely zeros must be written after the last word.

For example, to prepare for the calculation of the Montgomery parameter, ROS words are written for the modulus starting at address 0x1080 until address 0x1080+ROS-1. A word of all zeros must also be written at address 0x0180+ROS before starting the calculation.

13.4.1 Compute Montgomery parameter

During this operation the PKA computes the Montgomery parameter (R^2 mod n).

Table 35. Montgomery parameter computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	size (words)
Number of bits of the modulus	(In bits, 0 ≤ value < 4160)	0x0408	1
Modulus 'n'	(Odd integer only, n < 2^{4160})	0x1080	ROS

Table 36. Montgomery paramater computation outputs

Output data	Value (Note)	hexadecimal offset in PKA	size (words)
Montgomery parameter: (R^2 mod n)	-	0x0620	ROS

13.4.2 Compute modular exponentiation

During this operation the PKA computes the modular exponentiation (m^e mod n).

Table 37. Modular exponentiation computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Number of bits of the exponent 'e'	(in bits, not null)	0x0400	1
Number of bits of the operands 'm'	(in bits, not null)	0x0408	1
Storage area for Montgomery parameter. Must be used if MODE[5:0]="000010" (modular exponentiation only).	(mandatory)	0x0620	ROS
Base of the exponentiation 'm'	(0 ≤ m < n)	0x0C50	ROS
Exponent to process 'e'	(0 ≤ e < n)	0x0E60	ROS
Modulus 'n'	(Odd integer only, n < 2^{4160})	0x1080	ROS
Result (m^e mod n)	(0 ≤ result < n)	0x830	ROS

The execution time, for both operations, depends on the number of bits of the modulus and

exponent. When SECLVL=0, the execution time depends also on the value of the exponent. It is possible to make the execution time independent from the value of the exponent, by setting the SECLVL=1. To make the modular exponentiation operations constant time, the exponent size must be set equal to the size of the modulus, regardless of the exponent's actual number of significant bits.

Note: The core supports only odd modulus. if modulus randomization is applied, the randomized modulus has to be an odd number.

Both exponent and operand are unsigned integers represented in binary form on 64 bits. Both must be less or equal to max_rsa_size (4160)

13.4.3 Compute the ECC scalar multiplication

During this operation the PKA computes the ECC scalar multiplication kP.

Table 38. ECC scalar multiplication computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Number of bits of 'k'	(In bits, not null)	0x0400	1
Number of bits of the modulus GF(p). For NIST P-256 value is 256.	(In bits, not null, 8 < value < 640)	0x0408	1
ECC curve 'a' coefficient sign. 0x0: positive; 0x1: negative For NIST curves over prime fields value is 0x1.	– 0x0: positive – 0x1: negative	0x0410	1
ECC curve 'a' coefficient absolute value. For NIST curves over prime fields value is 0x3.	(absolute value, |a| < p)	0x0418	EOS
ECC curve prime modulus 'p'	(odd integer prime, 0 < p < 2^{640})	0x0470	EOS
Storage area for Montgomery Parameter. Must be used if MODE[5:0]="100010" (scalar multiplication only).	45	0x04C8	EOS
The 'k' of kP	(0 ≤ k < 2^{640})	0x0520	EOS
initial point 'P' coordinates X Y	 (x < p) (y < p)	 0x0578 0x05D0	 EOS EOS
Coordinates of the result X Y	 (result < p) (result < p)	 0x0578 0x05D0	 EOS EOS

The execution time, for both operations, depends on the size of the scalar. When SECLVL = 0, the execution time also depends on the value of the scalar.

It is possible to make the execution time independent from the value of the scalar, by setting the SECLVL equal to 01. To make the ECC scalar multiplication operations constant time, the size of the scalar must be set equal to the size of the order, regardless of the scalar's actual number of significant bits.

Note: The core supports only prime modulus; it is not possible to apply modulus randomization.

The input value k is required to be greater than zero.

If the user wants to execute the side channel protection known as scalar blinding (where the scalar 'k' is randomized via the addition of a multiple of the order of the curve n), the PKA core is not able to deal with the case where the internal temporary result falls in the point at infinity. It is very unlikely to fall in the point at infinity, but it is possible to check that this inconvenience has happened by checking the final result. I.e. if the temporary result is equal to the point at infinity, the final result has both coordinates x and y equal to zero, that is not a point on the curve. When the scalar 'k' is smaller than the order, it is not possible to reach the point at infinity.

13.4.4 Point check

During this operation the PKA computes a boolean that tell whether the given point P satisfies the curve over prime fields equation or not. If output error is equal to zero the point is on the curve.

Table 39. Point check computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Number of bits of the modulus GF(p). For NIST P-256 value is 256.	(In bits, not null, 8 < value < 640)	0x0408	1
ECC curve 'a' coefficient sign. 0x0: positive; 0x1: negative For NIST curves over prime fields value is 0x1.	0x0:positive 0x1:negative	0x0410	1
ECC curve 'a' coefficient absolute value.	(Absolute value, |a| < p)	0x0418	EOS
ECC curve 'b' coefficient absolute value.	(|b| < p)	0x0520	EOS
ECC curve prime modulus 'p'	(Odd integer prime, $0 < p < 2^{640}$)	0x0470	EOS
The point 'P' Coordinates X Y	 (x < *p*) (y < *p*)	 0x0578 0x05D0	2*EOS
Storage area for Montgomery parameter.	-	0x4C8	EOS

Table 40. Point check computation outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Error output result. If it is different from zero the point ‘P’ is not on the curve.	– 0xD60D: point on curve – 0xA3B7: point not on curve – 0xF946: x or y coordinate is not smaller than modulus p - 0xF09C: A fault has altered the result of operation - 0x0000: A fault has altered the execution of operation	0x0F80	1

If the error code does not correspond to any of the possible expected values, something wrong has happened during the execution flow.

The location storing the Error code is set equal to 0x400 at the beginning of the computation. If it is equal to 0x400 at the end of the computation, then it means that the fault check procedure has failed. This scenario should be treated as an attempt to skip the fault check procedure and thus as a fault itself.

Furthermore, the detection of faults is also signaled using dedicated flags (see *Section 13.3.5: PKA error management*).

Note: Coordinates X and Y of the point P must be smaller than the modulus.

13.4.5 ECDSA sign

During this operation the PKA computes a signed message using elliptic curves over prime fields.

Table 41. ECDSA sign computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Number of bits of the order For SHA-256 value is 256.	(in bits, not null)	0x0400	1
Number of bits of the modulus GF(p). For NIST P-256 value is 256.	(in bits, 8 < value < 640)	0x0408	1
ECC curve ‘a’ coefficient sign. 0x0: positive; 0x1: negative For NIST curves over prime fields value is 0x1.	0x0:positive 0x1:negative	0x0410	1
ECC curve ‘a’ coefficient absolute value. For NIST curves over prime fields value is 0x3.	(absolute value, |a| < p)	0x0418	EOS
ECC curve prime modulus ‘p’	(Odd integer prime, $0 < p < 2^{640}$)	0x0470	EOS
Random integer ‘k’ generated outside the PKA for this ECDSA signature	($0 \le k < 2^{640}$)	0x0520	EOS
The initial point ‘P’ Coordinates X Y	 (x < *p*) (y < p)	 0x0578 0x05D0	 EOS EOS
Hash of the message ‘e’	($e < 2^{M}$)	0x0E70	EOS
Private key ‘d’	(Positive integer)	0x0EC8	EOS
Integer prime order of the curve ‘n’	(Integer prime)	0x0F20	EOS

Table 42. ECDSA sign computation outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Error output result. If it is different from zero, a new ‘k’ should be generated and ECDSA sign operation needs to be repeated.	– 0xD60D: successful computation, no error – 0xCBC9: failed computation – 0xA3B7: signature part r is equal to 0 – 0xF946: signature part s is equal to 0 - 0x0000: A fault has altered the normal execution of the operation	0x0F80	1
Signature part ‘r’	192	0x0730	EOS
Signature part ‘s’	213	0x0788	EOS
The final point ‘kP’ (OPTIONAL) X Y	 783 804	 0x578 0x5D0	 EOS EOS

If the error code does not correspond to any of the possible expected values, something wrong has happened during the execution flow.

The location storing the Error code is set equal to 0 at the beginning of the computation. If it is equal to 0 at the end of the computation, then it means that the fault check procedure has failed. This scenario should be treated as an attempt to skip the fault check procedure and thus as a fault itself.

Note: The prime modulus for P-256 curve is p = 2^256 - 2^224 + 2^192 + 2^96 - 1 which correspond to hexadecimal value 0xFFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF.

Extended ECDSA support

PKA also supports extended ECDSA signature, for which the inputs and the outputs are the same as ECDSA signature, with the addition of the coordinates of the point kG. This extra output is defined below:

Table 43. Extended ECDSA outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Curve point kG coordinate x_1	($0 \le x_1 < p$)	0x0578	1
Curve point kG coordinate y_1	($0 \le y_1 < p$)	0x05D0	1

When SECLVL = 0, the execution time depends on the value of the scalar.

It is possible to make the execution time independent from the value of the scalar, by setting the security level equal to 1.

13.4.6 ECDSA verification

During this operation the PKA computes a boolean that tells whether given signature is valid or not. This signature shall be based on an elliptic curves over prime fields.

If output error is equal to zero the signature is valid.

Table 44. ECDSA verification computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Number of bits of the order. For SHA-256 value is 256.	(In bits, not null)	0x0408	1
Number of bits of the modulus GF(p). For NIST P-256 value is 256.	64 bits (In bits, not null, 8 < value < 640)	0x0F78	1
ECC curve ‘a’ coefficient sign. 0x0: positive; 0x1: negative For NIST curves over prime fields value is 0x1.	0x0:positive 0x1:negative	0x0410	1
ECC curve ‘a’ coefficient absolute value. For NIST curves over prime fields value is 0x3.	(Absolute value, |a| < p)	0x0418	EOS
ECC curve prime modulus ‘p’	(Odd integer prime, 0< p < 2^{640})	0x0F20	EOS
The initial point ‘P’ coordinates X Y	 (x < *p*) (y < *p*)	 0x0628 0x0680	2*EOS
Public key point ‘Qa’ coordinates X Y	 (x_Q < *p*) (y_Q < *p*)	 0x0730 0x0788	 EOS EOS
Signature part ‘r’	(0 < *r* < n)	0x0EC8	EOS
Signature part ‘s’	(0 < *s* < n)	0x08E8	EOS
Hash of the message ‘e’	(e < 2^M)	0x0E70	EOS
Integer prime order of the curve ‘n’	(integer prime)	0x0470	EOS

Table 45. ECDSA verification computation outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Error output result. If it is different from 0, the signature is not verified.	– 0xD60D: valid signature – 0xA3B7: invalid signature	0x0F80	1

13.4.7 RSA CRT exponentiation

During this operation the PKA computes the Chinese remainder theorem (CRT) optimization.

Table 46. RSA CRT exponentiation computation inputs

Input data	Value (Note)	Hexadecimal offset in PKA	Size (words)
Number of bits of the operands	(in bits, not null)	0x0408	1
CRT parameter 'd_p'	($0 \le d_p < 2^{M/2}$)	0x0728	ROS/2
CRT parameter 'd_q'	($0 \le d_q < 2^{M/2}$)	0x0E60	ROS/2
CRT parameter 'q_{inv}'	($0 \le q_{inv} < 2^{M/2}$)	0x0938	ROS/2
Prime 'p'	($0 \le P < 2^{M/2}$)	0x0B48	ROS/2
Prime 'q'	($0 \le q < 2^{M/2}$)	0x1080	ROS/2
Base of the exponentiation (A)	($0 \le A < 2^{M/2}$)	0x1290	ROS

Table 47. RSA CRT exponentiation computation outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result: $A^d \bmod pq$	($0 \le$ result $< pq$)	0x0830	ROS

Note: CRT is supported for modulus up to 4096 bits, with prime p and q of the same length i.e. half of the length of the modulus.

13.4.8 Modular reduction

During this operation the PKA computes a modular reduction.

Table 48. Modular reduction computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operand length	(In bits, not null)	0x0400	1
Operand A	($0 \le A < 2n < 2^{4160}$)	0x0408	1
Modulus length	(In bits, 8 < value < 4160)	0x0A40	1
Modulus n	(Odd integer only, $n < 2^{4160}$)	0x1080	ROS

Table 49. Modular reduction computation outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result A mod n	(0 < result < n)	0x0E60	ROS

13.4.9 Arithmetic addition

During this operation the PKA computes the arithmetic addition of two inputs op1 and op2.

Table 50. Arithmetic addition inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operand length M	(In bits, not null)	0x0408	1
op1 A	($0 \le A < 2^M$)	0x0A40	ROS
op2 B	($0 \le B < 2^M$)	0x0C50	ROS

Table 51. Arithmetic subtraction outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
operation result: op1 + op2	($0 \le$ result $< 2^{M+1}$)	0x0E60	ROS+1

13.4.10 Arithmetic subtraction

During this operation the PKA computes the arithmetic subtraction of two inputs op1 and op2.

Table 52. Arithmetic subtraction inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operand length M	(In bits, not null)	0x0408	1
op1 A	($0 \le A < 2^M$)	0x0A40	ROS
op2 B	($0 \le B < 2^M$)	0x0C50	ROS

Table 53. Arithmetic subtraction outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result: op1 - op2	($0 \le result < 2^M$)	0x0E60	ROS

13.4.11 Comparison

During this operation, given two inputs op1 and op2, the PKA computes comparisons as follow:

- If op1 = op2, then the result = 0
- If op1 > op2, then the result = 1
- If op1 < op2, then the result = 2

Table 54. Comparison computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operand length M	(In bits, not null)	0x0408	1
op1 A	($0 \leq A < 2^M$)	0x0A40	ROS
op2 B	($0 \leq B < 2^M$)	0x0C50	ROS

Table 55. Comparison computation outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result: If op1 = op2, then result = 0xED2C If op1 > op2, then result = 0x7AF8 If op1 < op2, then result = 0x916A Result = 0xF09C: A fault has altered the result of operation Result = 0x0000: A fault has altered the execution of operation	0xED2C, 0x7AF8, 0x916A, 0xF09C, 0x0000	0x0E60	1

If the error code does not correspond to any of the possible expected values, something wrong has happened during the execution flow.
The location storing the Error code is set equal to 0x400 at the beginning of the computation. If it is equal to 0x400 at the end of the computation, then it means that the fault check procedure has failed. This scenario should be treated as an attempt to skip the fault check procedure and thus as a fault itself.
Furthermore, the detection of faults is also signaled using dedicated flags (see *Section 13.3.5: PKA error management*)

13.4.12 Arithmetic multiplication

During this operation the PKA computes the arithmetic multiplication of two inputs op1 and op2.

Table 56. Arithmetic multiplication inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operand length M	(In bits, not null)	0x0408	1
op1 A	($0 \le A < 2^M$)	0x0A40	ROS
op2 B	($0 \le B < 2^M$)	0x0C50	ROS

Table 57. Arithmetic multiplication outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result: op1 x op2	($0 \le result < 2^M$)	0x0E60	2*ROS

13.4.13 Modular addition

During this operation the PKA computes the addition of op1 by op2 modulus op3.

Table 58. Modular addition inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operand length	(In bits, not null)	0x0408	1
op1 A	($0 \le A < n$)	0x0A40	ROS
op2 B	($0 \le B < n$)	0x0C50	ROS
op3 (modulus n)	($n < 2^{4160}$)	0x1080	ROS

Table 59. Modular addition outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result: (op1 + op2) mod op3	($0 \le result < n$)	0x0E60	ROS

With SECLVL=0, the execution time depends both on the size of the modulus and on whether the reduction after the addition is executed or not. To make the execution constant time (i.e., independent from the final reduction), security level shall be set to 1.

13.4.14 Modular inversion

During this operation the PKA computes the inversion of op1 modulus op2.

Note: op1 must be smaller than the modulus op2.

Table 60. Inverse op1 modulus op2 computation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operand length	(In bits, not null)	0x0408	1
op1 A	(0 ≤ A < n)	0x0A40	ROS
op2 (modulus n)	(Odd integer only, n < 2^{4160})	0x0C50	ROS

Table 61. Inverse op1 modulus op2 computation outputs

output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result A^{-1} mod n	0 < result < n	0x0E60	ROS

Note: The operand to invert should be coprime with the modulus. In case this is not known, and the operand is a divisor of the modulus, the result is a multiple of a factor of the modulus.

13.4.15 Modular subtraction

During this operation the PKA computes the subtraction of op1 by op2 modulus op3.

Table 62. Modular subtraction inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
opErand length	1	0x0408	1
op1 A	(0 ≤ A < n)	0x0A40	ROS
op2 B	(0 ≤ B < n)	0x0C50	ROS
op3 (modulus n)	(n < 2^{4160})	0x1080	ROS

Table 63. Modular subtraction outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result: (op1 - op2) mod op3	(0 ≤ result < n)	0x0E60	ROS

With SECLVL=0, the execution time depends both on the size of the modulus and on whether the reduction after the addition is executed or not. To make the execution constant time (that is, independent from the final reduction), security level shall be set to 1.

Note: Reduction happens when op2 is larger than op1. The result is always smaller than op3 and equal or larger than zero.

13.4.16 Montgomery multiplication

During this operation the PKA computes the multiplication of op1 by op2 modulus op3 in the Montgomery space.

Before issuing this operation the PKA needs to set some internal registers, which can be done by issuing an Montgomery parameter computation, an ECC scalar multiplication or any ECC operation.

Table 64. Montgomery multiplication inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operand length	(In bits, not null)	0x0408	1
op1 A	(0 ≤ A < n)	0x0A40	ROS
op2 B	(0 ≤ B < n)	0x0C50	ROS
op3 (modulus n)	(Odd integer only, n < 2^{4160})	0x1080	ROS

Table 65. Montgomery multiplication outputs

output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Operation result (AxB mod n)	-	0x0E60	ROS

Note: For converting a number from natural representation to Montgomery domain it is necessary to perform a Montgomery multiplication between the number and the Montgomery Parameter (called also R2 mod n). While for converting a number from Montgomery domain to natural domain a Montgomery multiplication between the Montgomery number and one has to be computed.

13.4.17 ECC complete addition

ECC complete addition computes the addition of two given points on an elliptic curve. Operation instructions are summarized below.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x,y) conversion (X,Y,Z) = (x,y,1) can be used. To convert resulting point to Jacobian coordinates conversion (x,y)= (X/Z^2,Y/Z^3) can be used.

Table 66. ECC complete addition inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Curve modulus *p* length	(In bits, not null, 8 < value < 640)	0x0408	1
Curve coefficient *a* sign	0x0:positive 0x1:negative	0x0410	1

Table 66. ECC complete addition inputs (continued)

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Curve modulus value *p*	(Odd integer prime, 0 < p < 2^{640})	0x0470	EOS
Curve coefficient |*a*|	(Absolute value, |a| < p)	0x0418	EOS
First point P coordinate X	(x < *p*)	0x0628	EOS
First point P coordinate Y	(y < *p*)	0x0680	EOS
First point P coordinate Z	(z < *p*)	0x06D8	EOS
Second point Q coordinate X	(x < *p*)	0x0730	EOS
Second point Q coordinate Y	(y < *p*)	0x0788	EOS
Second point Q coordinate Z	(z < *p*)	0x07E0	EOS

Table 67. ECC complete addition outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Result coordinate X	(x < *p*)	0x0D60	EOS
Result coordinate Y	(y < *p*)	0x0DB8	EOS
Result coordinate Z	(z < *p*)	0x0E10	EOS

13.4.18 ECC double base ladder

ECC double base ladder operation consists in the computation of k*P+m*Q, where (P,Q) are two points on an elliptic curve and (k,m) are two scalars. Operation instructions are summarized below. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y, Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x, y)

Table 68. ECC double base ladder operation inputs

Input data	Value (note)	Hexadecimal offset in PKA	Size (words)
Curve prime order *n* length	(In bits, not null)	0x0400	1
Curve modulus *p* length	(In bits, not null, 8 < value < 640)	0x0408	1
Curve coefficient *a* sign	0x0:positive 0x1:negative	0x0410	EOS
Curve coefficient |*a*|	(Absolute value, |a| < p)	0x0418	EOS
Curve modulus value *p*	(odd integer prime, 0 < p < 2^{640})	0x0470	EOS
Integer k	(odd integer prime, 0 < k < 2^{640})	0x0520	EOS
Integer m	(odd integer prime, 0 < m < 2^{640})	0x0578	EOS
First point P coordinate X	(x < *p*)	0x0628	EOS
First point P coordinate Y	(y < *p*)	0x0680	EOS
Second point Q coordinate X	(x < *p*)	0x0730	EOS
Second point Q coordinate Y	(y < *p*)	0x0788	EOS

Table 69. ECC double base ladder operation outputs

Output data	Value (note)	Hexadecimal offset in PKA	Size (words)
Result coordinate X	(x < *p*)	0x0578	EOS
Result coordinate Y	(y < *p*)	0x05D0	EOS
Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	0x0F80	EOS

13.5 Processing time

The following tables summarize the PKA computation times, expressed in clock cycles.

Table 70. Modular exponentiation(1)

Exponent length (in bits)	Mode	Operand length (in bits)			
		1024	2048	3072	4096
3	Normal	124600	491000	684000	1133200
	Fast	22700	82000	178000	311000
17	Normal	135700	531400	772400	1288000
	Fast	33800	122500	266500	465800
$2^{16} + 1$	Normal	180000	693700	1126200	1907200
	Fast	78200	284700	620400	1085000
1024	Normal	5850000	-	-	-
	Fast	5748000	-	-	-
	SECLVL=1	7579385	-	-	-
2048	Normal	-	42240000	-	-
	Fast	-	41832000	-	-
	SECLVL=1	-	55409889	-	-
3072	Normal	-	-	136830000	-
	Fast	-	-	136325000	-
	SECLVL=1	-	-	181240525	-
4096	Normal	-	-	-	316000000
	Fast	-	-	-	315226000
	SECLVL=1	-	-	-	422820009

1. To make the modular exponentiation operations constant time, the exponent size must be set equal to the size of the modulus, regardless of the exponent's actual number of significant bits.

Table 71. ECC scalar multiplication(1)

Mode	Modulus length (in bits)							
	160	192	256	320	384	512	521	640
Normal	-	1590000	3083000	5339000	8518000	17818000	21053000	31826000

1. These times depend on the number of "1"s included in the scalar parameter.

Table 72. ECDSA signature average computation time(1)(2)

Modulus length (in bits)								
SECLVL	**160**	**192**	**256**	**320**	**384**	**512**	**521**	**640**
0	-	1500000	2744000	4579000	7184000	14455000	16685000	24965000
1	-	1902700	3709700	6311600	10088100	20947900	23997500	37177500

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 73. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
1011000	1495000	2938000	5014000	7979000	16804000	19254000	29582000

Table 74. Montgomery parameters average computation times

Modulus length (in bits)									
160	**192**	**256**	**320**	**384**	**512**	**521**	**1024**	**2048**	**3072**
2259	3923	5924	7451	10841	17506	32000	59768	233073	552321

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

13.5.1 PKA interrupts

There are three individual maskable interrupt sources generated by the public key accelerator when enabled by EN bit, signaling the following events:

1. Access to unmapped address (ADDRERRF), see *Section 13.3.5: PKA error management*
2. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 13.3.5: PKA error management*
3. PKA end of operation (PROCENDF), see *Section 13.3.5: PKA error management*
4. PKA FSM fault (FAULTFSMF), see *Section 13.3.5: PKA error management*
5. PKA ERRORCODE fault (FAULTERRORCODEF), see *Section 13.3.5: PKA error management*

The three interrupt sources are connected to the same global interrupt request signal pka_it. The user can enable or disable above interrupt sources individually by changing the mask bits in the *Section 13.6.1: PKA control register (PKA_CR)*.Setting the appropriate mask bit to 1 enables the
interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register, except for fault flags which can be cleared only putting low EN bit in PKA_CR.

Table 75 gives a summary of the available features.

Table 75. PKA interrupt requests

Interrupt event	Event flag	Enable control bit
Access to unmapped address error	ADDRERRF	ADDRERRIE
PKA RAM access error	RAMERRF	RAMERRIE
PKA end of operation	PROCENDF	PROCENDIE
PKA FSM fault	FAULTFSMF	FAULTFSMIE
PKA ERROR code fault	FAULTERRORCODEF	FAULTERRORCODIE

13.6 PKA registers

13.6.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FAULTERRORCODEIE	FAULTFSMIE	Res.	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
								rw	rw		rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE						Res	Res	Res	Res	Res.	SECLVL	START	EN
		rw	rw	rw	rw	rw	rw						rw	rw	rw

Bits 31:24 Reserved, must be kept at zero

Bits 23 **FAULTERRORCODEIE** : Fault Error code interrupt enable
0: Interrupt is disabled.
1: An interrupt is generated when FAULTERRORCODEF (PKA_SR[23] is set.

Bits 22 **FAULTFSMIE** : Fault FSM interrupt enable
0: Interrupt is disabled.
1: An interrupt is generated when FAULTFSMF (PKA_SR[22] is set.

Bit 21 Reserved, must be kept at zero

Bit 20 **ADDRERRIE** : Address error interrupt enable
0: Interrupt is disabled.
1: An interrupt is generated when ADDRERRF (PKA_SR[20] is set.

Bit 19 **RAMERRIE** : RAM error interrupt enable
0: Interrupt is disabled.
1: An interrupt is generated when RAMERRF (PKA_SR[19]) is set.

Bit 18 Reserved, must be kept at zero

Bit 17 **PROCENDIE** : End of operation interrupt enable
0: Interrupt is disabled.
1: An interrupt is generated when PROCENDF (PKA_SR[17]) is set.

Bits 16:14 Reserved, must be kept at zero

Bits 13: 8 **MODE[5:0]**: PKA operation code

000000 : *Compute Montgomery parameter and modular exponentiation*
000001 : *Compute Montgomery parameter*
000010 : *Compute modular exponentiation only (Montgomery parameter should be loaded)*
100000 : *Compute Montgomery parameter and compute ECC kP operation*
100010 : *Compute the ECC kP primitive only (Montgomery parameter should be loaded)*
100011 : *ECC complete addition*
100100 : *ECDSA sign*
100110 : *ECDSA Verification*
100111: *ECC double base ladder*
101000 : Point Check
000111 : *RSA CRT exponentiation*
001000 : *Modular inversion*
001001 : *Arithmetic addition*
001010 : *Arithmetic Subtraction*
001011 : *Arithmetic multiplication*
001100 : *Comparison*
001101 : *Modular Reduction*
001110 : *Modular Addition*
001111 : *Modular Subtraction*
010000 : *Montgomery Multiplication*

Bit 7:3 Reserved, must be kept at zero

Bit 2 **SECLVL** : Security enable.

0: No constant time operation

1: Constant time operation

Bit 1 **START** : start the operation

Writing ‘1’ to this bit starts the operation which is selected by MODE[5:0], using the operands and data already written to the PKA RAM. This bit is always read as ‘0’.

Nota: START is ignored if PKA is busy.

Bit 0 **EN** : Peripheral enable.

0 : Disable PKA.

1 : Enable PKA.

13.6.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FAULTERRORCODEF	FAULTFSMF	Res.	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY
								rw	rw		r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res

Bits 31:24 Reserved, must be kept at zero

Bits 23 **FAULTERRORCODEF** : Fault Error code flag
0: No Fault error code
1: Fault error code has been generated

Bits 22 **FAULTFSMF** : Fault FSM flag
0: No Fault FSM error
1: Fault FSM error has been generated

Bit 21 Reserved, must be kept at zero

Bit 20 **ADDRERRF**: Address error flag
0: No Address error
1: Address access is out of range (unmapped address)

Bit 19 **RAMERRF**: PKA RAM error flag
0: No PKA RAM access error
1: An AHB access to the PKA RAM occured while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

Bits 18 Reserved, must be kept at zero

Bit 17 **PROCENDF**: PKA End of Operation flag
0: Operation in progress
1: PKA operation is completed. This flag is set when the BUSY bit is de-asserted.

Bit 16 **BUSY**: PKA operation is in progress
This bit is set to ‘1’ whenever START bit in the PKA_CR is set. It is automatically cleared when the computation is complete, meaning that PKA RAM can be safely accessed and a new operation can be started.
0: No operation is in progress (default)
1: An operation is in progress
Note: if PKA is started with a wrong opcode the IP is busy for a couple of cycles then it aborts automatically the operation and go back to ready (BUSY bit is set to ‘0’).

Bits 15:0 Reserved, must be kept at zero

13.6.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDR ERRFC	RAM ERRFC	Res.	PROC ENDFC	Res.
											w_r0	w_r0		w_r0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.

Bits 31 : 21 Reserved, must be kept at zero

Bit 20 **ADDRERRFC**: Clear Address error flag
0: No action
1: Clear the ADDRERRF flag

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
0: No action
1: Clear the RAMERRF flag

Bits 18 Reserved, must be kept at zero

Bit 17 **PROCENDFC**: Clear PKA End of Operation flag
0: No action
1: Clear the PROCENDF flag

Bits 16 : 0 Reserved, must be kept at zero

Note: Reading PKA_CLRFR returns all zeros.

13.6.4 PKA RAM memory

Address offset: 0x400

The PKA_RAM is memory mapped at the offset address of 0x0400 compared to the PKA base address. Only word access (64 bits) are supported.

RAM size is 4768 bytes (max word offset: 0x1698)

13.6.5 PKA register map

Table 76. PKA register map

Offset	Register	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
0x0000	**PKA_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5 : 0]											SCLVL	START	EN
	Reset value	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
0x0004	**PKA_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	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
0x0008	**PKA_CLRFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	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

14 True random number generator (TRNG)

14.1 Overview

The true random number generator (TRNG) is a hardware module able to generate a random sequence of bits.

The TRNG is based on an analog source of entropy composed by free 9 running ring oscillators. The outputs of those oscillators are XORed and sampled for providing random bits which are then processed by a digital stage.

The digital stage implements an AES-CMC core and is compliant with NIST SP800-90B specifications.

14.2 Features

The TRNG has the following features:

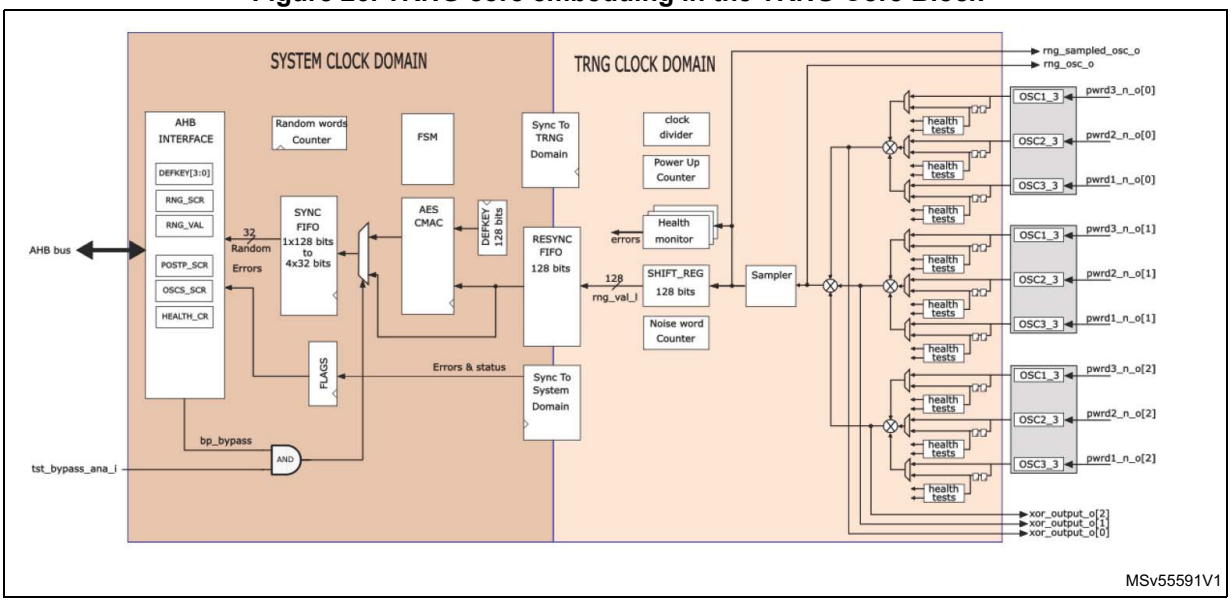
- Deliver 128-bit random number, produced by an analog generator
- Generation of true random values based on a dedicated analog source of entropy
- SP800-90B post processing based on 128-bit AES-CMAC
- Programmable default key for AES
- Continuous Health tests for controlling random source behavior
- On-demand test of analog source and post-processing
- Enabling/disabling of each analog oscillator for power saving
- Hardware check of sampling clock
- AHB slave peripheral. All IP registers must be accessed using 32-bit read or write access. An error response is provided if access size is not 32 bits.
- Internal four 32-bit FIFO for storing random data
- Internal clock enable divider for lowering sampling clock frequency.
- 2 clock domains:
  - *hclk_i* clock for the AHB interface
  - rngclk_i clock for sampling and shifting bits; frequency must be between 2 to 255 times lower than *hclk_i* one and with a duty cycle of 50%.
- 128*N cycles of rngclk_i clock plus 220 cycles of *hclk_i* between two consecutive random number generations where N is in range of 1 to 16.

14.3 TRNG block diagram

The TRNG design includes an AHB Bus interface implementing configuration and control registers, an output FIFO, the analog source of entropy and the digital stage of post processing.

Figure 26 describes the TRNG core embedded inside the TRNG Core Block.

Figure 26. TRNG core embedding in the TRNG Core Block



14.4 TRNG functional description

The slave interface includes all the TRNG memory mapped registers.

The CPU enables the TRNG by resetting bit DISABLE of TRNG_CR unless already enabled (default is TRNG enabled after reset). TRNG startup always begins with health tests of the analog source and the digital post processing.

After the TRNG IP is disabled, by setting CR.DISABLED, in order to properly restart the TRNG IP, the AES_RESET bit must be set to 1 (that is, resetting the AES core and restarting all health tests).

If any error is detected during the health test step, TRNG stops and set errors flags in RNG_SR and RNG_POSTP_SR registers. If no error is found, then TRNG generates random that are used for initializing the AES-based post processing.

When initialization is completed, TRNG starts generating random numbers. TRNG is continuously running and valid Random values are stored in an internal four 32-bit FIFO. It is possible to force a new TRNG reinitialization after generation of a specific number of random numbers as defined in register RNG_POSTP_CR.

CPU must poll FIFO status bits FIFO_FULL and VAL_READY in TRNG_SR before reading random values.

TRNG includes an internal clock enable divider that can be set by bits CLKDIV0 to CLKDIV15 of TRNG_CR register. It is used for decreasing internal random source sampling. Before changing the clock divider, it is recommended to disable the TRNG bit setting bit DISABLE to 1. Due to resynchronization latency between system clock domain and TRNG core clock domain, the value of DISABLE bit reflects the written value when it has been seen by the TRNG core clock domain.

Note: To use the TRNG peripheral the system clock frequency must be at least 32 MHz

For the same reasons, a new divider value can be written provided that both clock domains have the same old value. While the new divider value is being resynchronized with the TRNG core clock domain, it is not possible to write another new value. Consequently, the user must read the RNG_CR register and check that the divider value is equal to the one it has just written. If not, it must redo the write and checking.

Error flags in TRNG_SR reflect the current status of the TRNG core and not the random values stored in the FIFO. No value is stored if TRNG core is in error.

Health tests of analog noise source run continuously but contrary to initialization step, an error detection does not stop random generation but only sets error flags. These error flags can be reset by writing bit RST_HEALTH_FLAGS of TRNG_CR register. When set, this bit remains at 1 until its value has been seen in TRNG core clock domain. Then it is automatically reset.

Heath test of digital post processing can be triggered on-demand by setting bit AES_RESET of RNG_POSTP_CR. When set, this bit remains at 1 until its value has been seen in TRNG core clock domain. Then it is automatically reset. This also triggers the restart of health tests of analog noise source.

As a general comment, any modification of control and configuration bits of TRNG Control registers that impact TRNG core applies in TRNG clock domain after some latency inherent to clock domain resynchronization.

Similarly, any flag or result generated in TRNG clock domain is seen with some latency in system clock delay. However, this latency is shorted as system clock frequency is much higher than TRNG one.

14.5 TRNG register access

The TRNG can only be accessed using 32-bit (word) accesses. References using a different size are invalid.

14.6 Memory map

The TRNG memory map is shown in the following table.

Table 77. TRNG memory map and address offsets

Offset	Register description (register name)
0x000	*Core Control Register (TRNG_CR)*
0x004	*Core Status Register (TRNG_SR)*
0x008	*32-bit Random Value (TRNG_VAL)*
0x030	*Oscillator Control Register (TRNG_OSCS_CR)*
0x034	*Post Processing Control Register (TRNG_POSTP_CR)*
0x038	*Post Processing Status Register (TRNG_POSTP_SR)*
0x040	*Bits 31 to 0 of AES 128-bit Default Key (TRNG_DEFKEY0)*
0x044	*Bits 63 to 32 of AES 128-bit Default Key (TRNG_DEFKEY1)*
0x048	*Bits 95 to 64 of AES 128-bit Default Key (TRNG_DEFKEY2)*
0x04C	*Bits 127 to 96 of AES 128-bit Default Key (TRNG_DEFKEY3)*
0x060	*Health Test Control Register (TRNG_HEALTH_CR)*
0x068	*OSC1 Health Tests Control Register (TRNG_HEALTH_OSC1_CR)*
0x06C	*OSC2 Health Tests Control Register (TRNG_HEALTH_OSC2_CR)*
0x070	*OSC3 Health Tests Control Register (TRNG_HEALTH_OSC3_CR)*
0x074	*OSC1 Health Tests Status Register (TRNG_HEALTH_OSC1_SR)*
0x078	*OSC2 Health Tests Status Register (TRNG_HEALTH_OSC2_SR)*
0x07C	*OSC3 Health Tests Status Register (TRNG_HEALTH_OSC3_SR)*

Table 77. TRNG memory map and address offsets

Offset	Register description (register name)
0x080	*TRNG Interrupt Control Register (TRNG_IRQ_CR)*
0x084	*Interrupt Status Register (TRNG_IRQ_SR)*

14.7 TRNG registers

The following sections detail the individual registers within the TRNG programming model.

14.7.1 Core Control Register (TRNG_CR)

Address: BaseAddress + 0x000

Type: R/W

Reset value: 0x0000_FF00

Description: Core Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLKDIV							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLKDIV								RST_HEALTH_FLAGS	CLR_REVCLK_FLAG	Res.	Res.	Res.	Res.	Res.	DISABLE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:8 **CLKDIV**: Sampling Clock Enable Divider.

CLKDIV[15:0] control the sampling clock enable divider, dividing by a factor equal to CLKDIV[15:0] + 1, values being in the range of 1 to 65536.

NB: A new value can be set if current value is visible by both clock domains otherwise write fails.

Bit 7 **RST_HEALTH_FLAGS**: Reset Health error flags when writing a “1” without resetting the whole TRNG.

0: no reset

1: reset health flag

When writing a 1, the value remains until it is seen by RNG core clock domain after resynchronization. Then it is automatically reset.

Bit 6 **CLR_REVCLK_FLAG**: Clear reveal clock error flags when writing a “1” without resetting the whole TRNG. 0: no reset

1: Clear revclk flag

Bits 5:1 Reserved, must be kept at reset value.

Bit 0 **DISABLE**:

Bit DISABLE can be used for reading or setting the state of the TRNG core. The value read is always the one available at the rng core clock domain.

0: The RNG core is enabled 1: The RNG core is disabled

When changing the value, the change is effective when the value read is the same as the one written.

14.7.2 Core Status Register (TRNG_SR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OSCS_ADAPT_ERROR	OSCS_REPET_ERROR	OSCS_HEALTH_DONE	ADAPT_ERROR	REPET_ERROR	SRC_HEALTH_DONE	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	r	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFO_FULL	VAL_READY	ENTROPY_ERR	REVEAL_CLK_ERR	ALL_OSCS_DOWN	TRNG_DISABLED
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Address: BaseAddress + 0x004

Type: R/W

Reset value: 0x0

Description: Core Status Register

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **OSCS_ADAPT_ERROR**:
Logical OR of adaptive health test errors of individual oscillators composing the noise source.

Bit 24 **OSCS_REPET_ERROR**: Logical OR of repetition health test errors of individual oscillators composing the noise source.

Bit 23 **OSCS_HEALTH_DONE**:
First run of source health tests of individual oscillators composing the noise source are completed.

Bit 22 **ADAPT_ERROR**: Noise source Adaptive health test error

Bit 21 **REPET_ERROR**: Noise source Repetition health test error

Bit 20 **SRC_HEALTH_DONE**: First run of noise source health test is completed

Bits 19:6 Reserved, must be kept at reset value.

Bit 5 **FIFO_FULL**: Indicates whether random data FIFO is full. 0: FIFO is not full.
1: The internal data FIFO is full and four 32:bit random values can be read.

Bit 4 **VAL_READY**: TRNG Value ready
At least one 32-bit random value is available in the data FIFO. Note that application must ensure that a random is available in internal FIFO by reading the VAL_READY flag before starting a read, otherwise a null value is returned.
0: No value is ready in FIFO.
1: A 32-bit value is available in the internal FIFO

Bit 3 **ENTROPY_ERR**: The error reported refers to a fault in the bit sequence detected by the Entropy Monitor. Failed test is given by REPET_ERROR, ADAPT_ERROR, OSCS_REPET_ERROR and OSCS_ADAPT_ERROR status flags.
0: No fault detected.
1: Embedded heath monitor detects an error in bit stream quality

Bit 2 **REVEAL_CLK_ERR**: The internal clock for the RNG core is not revealed. 0: Internal clock for RNG clock is present.
1: Internal RNG clock is not present.

Bit 1 **ALL_OSCS_DOWN**: All oscillators of the random source noise have been powered down. This can cause the rising of OEC3 flag.
1: All oscillators are down
0: At least one oscillator is ON

Bit 0 **TRNG_DISABLED**: TRNG is disabled.
0: Normal operation.
1: RNG is disabled.

14.7.3 32-bit Random Value (TRNG_VAL)

Address: BaseAddress + 0x008

Type: R

Reset value: 0x0

Description: 32-bit Random Value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RND_VAL															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RND_VAL															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RND_VAL**: a 32-bit Random Value. This is the output of the internal four-word FIFO.
Note that application must ensure that a random value is available in the FIFO by reading the VAL_READY flag before starting a read, otherwise a null value is returned.

14.7.4 Oscillator Control Register (TRNG_OSCS_CR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYNC_OSCS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	PWRD3			PWRD2			PWRD1			Res.
r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Address: BaseAddress + 0x030

Type: R/W

Reset value: 0x8000_0000

Description: OscillatorControl Register

Bit 31 **SYNC_OSCS**: When set, selection of resynchronized output of oscillators.
NB: It is recommended to disable TRNG before changing this value.

Bits 30:10 Reserved, must be kept at reset value.

Bits 9:7 **PWRD3**: Power down of individual oscillators in triple-oscillator block number 3

Bits 6:4 **PWRD2**: Power down of individual oscillators in triple-oscillator block number 2

Bits 3:1 **PWRD1**: Power down of individual oscillators in triple-oscillator block number 1

Bit 0 Reserved, must be kept at reset value.

14.7.5 Post Processing Control Register (TRNG_POSTP_CR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NB_RND_REINIT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NB_LOOP_AES				Res.	Res.	Res.	Res.	Res.	Res.	Res.	AES_RESET
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	w

Address: BaseAddress + 0x034

Type: R/W

Reset value: 0x0000_0F00

Description: AES-based post-processing Control Register

Bits 31:16 **NB_RND_REINIT**: Number of 128-bit random words generated before AES automatically resets. This number is in the range of 1 to 65535 words. Value 0x0000 means that AES is never reinitialized.

NB: It is recommended to disable TRNG before changing this value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NB_LOOP_AES**: the number of 128-bit words got from the noise source that have to be processed by AES for generating a single 128-bit random word.

By default, this value is set to 2 (128 bits generated before an AES processing). 0 value means 16 loops.

A new AES processing is started only when the previous one is completed. NB: It is recommended to disable TRNG before changing this value.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **AES_RESET**: Reset AES post processing.

When writing a 1, the AES post processing is reinitialized, resulting in a new key and new state generation before 128-bit random words generation. The ‘1’ written is frozen until it is seen by RNG core clock domain after resynchronization. Then it is automatically reset.

It also reruns analog source health tests.

0: No effect

1: Reset AES core

14.7.6 Post Processing Status Register (TRNG_POSTP_SR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AES_DOUT_ERROR	AES_K12_ERROR	AES_HEALTH_DONE	AES_BUSY	AES_KEY_LD	AES_INIT	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Address:BaseAddress + 0x038

Type:R/W

Reset value:0x0000_0000

Description:AES-based Post Processing Status Register.

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **AES_DOUT_ERROR**: Health test error on AES-CMAC output generation

Bit 5 **AES_K12_ERROR**: Health test error on AES-CMAC sub-keys generation

Bit 4 **AES_HEALTH_DONE**: AES-CMAC health test is completed

Bit 3 AES_BUSY: AES core is busy, generating a random value. 0: AES core is idle
1: AES core is busy

Bit 2 **AES_KEY_LD**: AES random key has been generated and loaded in AES key register.
0: AES core is waiting for 128 random bits from the entropy sources for generating its key 1: AES key register has been loaded with a random key

Bit 1 **AES_INIT**: AES Post processing has been fully initialized (key and state) and is ready for generating 128- bit random words.
0: AES core is not initialized (no key or state set). 1: AES core is fully initialized.

Bit 0 Reserved, must be kept at reset value.

14.7.7 Bits 31 to 0 of AES 128-bit Default Key (TRNG_DEFKEY0)

Address: BaseAddress + 0x040

Type:R/W

Reset value:0xFFFF FFFF

Description:First part of 128-bit AES Default Key. This register is write-only as it can hold a device-specific key that masks the random key generated from the entropy source.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNG_DEFKEY0															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_DEFKEY0															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RNG_DEFKEY0**: Bits 31 to 0 of AES 128-bit Default Key

14.7.8 Bits 63 to 32 of AES 128-bit Default Key (TRNG_DEFKEY1)

Address:BaseAddress + 0x044

Type:R/W

Reset value: 0xFFFF FFFF

Description: Second part of 128-bit AES Default Key. This register is write-only as it can hold a device-specific key that masks the random key generated from the entropy source.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNG_DEFKEY1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_DEFKEY1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RNG_DEFKEY1**: Bits 63 to 32 of AES 128-bit Default Key

14.7.9 Bits 95 to 64 of AES 128-bit Default Key (TRNG_DEFKEY2)

Address: BaseAddress + 0x048

Type: R/W

Reset value: 0xFFFF FFFF

Description: Third part of 128-bit AES Default Key. This register is write-only as it can hold a device-specific key that masks the random key generated from the entropy source.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNG_DEFKEY2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_DEFKEY2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RNG_DEFKEY2**: Bits 95 to 64 of AES 128-bit Default Key

14.7.10 Bits 127 to 96 of AES 128-bit Default Key (TRNG_DEFKEY3)

Address: BaseAddress + 0x04C

Type: R/W

Reset value: 0xFFFF FFFF

Description: Fourth and last part of 128-bit AES Default Key. This register is write-only as it can hold a device-specific key that masks the random key generated from the entropy source.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNG_DEFKEY3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_DEFKEY3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RNG_DEFKEY3**: Bits 127 to 96 of AES 128-bit Default Key

14.7.11 Health Test Control Register (TRNG_HEALTH_CR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ITER_ADAP		Res.	Res.	ADAP_CUTOFF									
r	r	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPET_CUTOFF.							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Address: BaseAddress + 0x060

Type: R/W

Reset value: 0x02BB 0033

Description: The TRNG_HEALTH_CR register controls the health test parameters. It must be used with care as bad settings could prevent TRNG to work properly. The default values are valid for all ring oscillators enabled. When powering down some oscillators, the cutoff values must be increased as health tests could trigger an error. When all oscillators but one are powered down, the values should be the same as the ones defined in TRNG_HEALTH_OSCx_CR registers.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ITER_ADAP**: Number of iterations minus 1 of Adaptive test during initialization phase. Default value is set to 0 i.e. 1 iteration.
NB: It is recommended to disable TRNG before changing this value.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **ADAP_CUTOFF**: Cutoff value of Adaptive Test. The default value is set to 699.
Caution: To be handled with care as any change can lead to misbehavior of TRNG. NB: It is recommended to disable TRNG before changing this value.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REPET_CUTOFF**: Cutoff value of Repetition Test. The default value is set to 51.
Caution: To be handled with care as any change can lead to misbehavior of TRNG. NB: It is recommended to disable TRNG before changing this value.

14.7.12 OSC1 Health Tests Control Register (TRNG_HEALTH_OSC1_CR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	ADAP_CUTOFF_OSC1									
r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPET_CUTOFF_OSC1							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Address: BaseAddress + 0x068

Type: R/W

Reset value: 0x03E3 00FB

Description: The TRNG_HEALTH_CR register controls the health test parameters of first oscillator in triple-oscillator cells. It must be used with care as bad settings could prevent TRNG to work properly.

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **ADAP_CUTOFF_OSC1**: Cutoff value of Adaptive Test for first oscillator of triple-oscillator cells. The default value is set to 995.
Caution: To be handled with care as any change can lead to misbehavior of TRNG. NB: It is recommended to disable TRNG before changing this value.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REPET_CUTOFF_OSC1**: Cutoff value of Repetition Test for first oscillator of triple-oscillator cells. The default value is set to 251.
Caution: To be handled with care as any change can lead to misbehavior of TRNG. NB: It is recommended to disable TRNG before changing this value.

14.7.13 OSC2 Health Tests Control Register (TRNG_HEALTH_OSC2_CR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	ADAP_CUTOFF_OSC2									
r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPET_CUTOFF_OSC2							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Address: BaseAddress + 0x06C

Type: R/W

Reset value: 0x03E3 00FB

Description: The TRNG_HEALTH_CR register controls the health test parameters of second oscillator in triple-oscillator cells. It must be used with care as bad settings could prevent TRNG to work properly.

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **ADAP_CUTOFF_OSC2**: Cutoff value of Adaptive Test for second oscillator of triple-oscillator cells. The default value is set to 995.

Caution: To be handled with care as any change can lead to misbehavior of TRNG. NB: It is recommended to disable TRNG before changing this value.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REPET_CUTOFF_OSC2**: Cutoff value of Repetition Test for second oscillator of triple-oscillator cells. The default value is set to 251.

Caution: To be handled with care as any change can lead to misbehavior of TRNG. NB: It is recommended to disable TRNG before changing this value.

14.7.14 OSC3 Health Tests Control Register (TRNG_HEALTH_OSC3_CR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	ADAP_CUTOFF_OSC3									
r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPET_CUTOFF_OSC3							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Address: BaseAddress + 0x70

Type: R/W

Reset value: 0x03E3 00FB

Description: The TRNG_HEALTH_CR register controls the health test parameters of third oscillator in triple-oscillator cells. It must be used with care as bad settings could prevent TRNG to work properly.

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **ADAP_CUTOFF_OSC3**: Cutoff value of Adaptive Test for third oscillator of triple-oscillator cells. The default value is set to 995.
Caution: To be handled with care as any change can lead to misbehavior of TRNG. NB: It is recommended to disable TRNG before changing this value.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REPET_CUTOFF_OSC3**: Cutoff value of Repetition Test for third oscillator of triple-oscillator cells. The default value is set to 251.
Caution: To be handled with care as any change can lead to misbehavior of TRNG. NB: It is recommended to disable TRNG before changing this value.

14.7.15 OSC1 Health Tests Status Register (TRNG_HEALTH_OSC1_SR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO3_ADAPT_ERROR	TO3_REPET_ERROR	TO2_ADAPT_ERROR	TO2_REPET_ERROR	TO1_ADAPT_ERROR	TO1_REPET_ERROR
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Address:BaseAddress + 0x074

Type: R/W

Reset value: 0x0

Description: OSC1 Health Tests Status Register

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **TO3_ADAPT_ERROR**: Adaptive error flag of first oscillator of third triple-oscillator cell.

Bit 4 **TO3_REPET_ERROR**: Repetition error flag of first oscillator of third triple-oscillator cell.

Bit 3 **TO3_ADAPT_ERROR**: Adaptive error flag of first oscillator of second triple-oscillator cell.

Bit 2 **TO2_REPET_ERROR**: Repetition error flag of first oscillator of second triple-oscillator cell.

Bit 1 **TO1_ADAPT_ERROR**: Adaptive error flag of first oscillator of first triple-oscillator cell

Bit 0 **TO1_REPET_ERROR**: Repetition error flag of first oscillator of first triple-oscillator cell.

14.7.16 OSC2 Health Tests Status Register (TRNG_HEALTH_OSC2_SR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO3_ADAPT_ERROR	TO3_REPET_ERROR	TO2_ADAPT_ERROR	TO2_REPET_ERROR	TO1_ADAPT_ERROR	TO1_REPET_ERROR
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Address: BaseAddress + 0x078

Type: R/W

Reset value: 0x0

Description: OSC1 Health Tests Status Register

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **TO3_ADAPT_ERROR**: Adaptive error flag of second oscillator of third triple-oscillator cell.

Bit 4 **TO3_REPET_ERROR**: Repetition error flag of second oscillator of third triple-oscillator cell.

Bit 3 **TO3_ADAPT_ERROR**: Adaptive error flag of second oscillator of second triple-oscillator cell.

Bit 2 **TO2_REPET_ERROR**: Repetition error flag of second oscillator of second triple-oscillator cell.

Bit 1 **TO1_ADAPT_ERROR**: Adaptive error flag of second oscillator of first triple-oscillator cell

Bit 0 **TO1_REPET_ERROR**: Repetition error flag of second oscillator of first triple-oscillator cell.

14.7.17 OSC3 Health Tests Status Register (TRNG_HEALTH_OSC3_SR)

Address: BaseAddress + 0x07C

Type: R/W

Reset value: 0x0

Description: OSC3 Health Tests Status Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO3_ADAPT_ERROR	TO3_REPET_ERROR	TO2_ADAPT_ERROR	TO2_REPET_ERROR	TO1_ADAPT_ERROR	TO1_REPET_ERROR
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **TO3_ADAPT_ERROR**: Adaptive error flag of third oscillator of third triple-oscillator cell.

Bit 4 **TO3_REPET_ERROR**: Repetition error flag of third oscillator of third triple-oscillator cell.

Bit 3 **TO3_ADAPT_ERROR**: Adaptive error flag of third oscillator of second triple-oscillator cell.

Bit 2 **TO2_REPET_ERROR**: Repetition error flag of third oscillator of second triple-oscillator cell.

Bit 1 **TO1_ADAPT_ERROR**: Adaptive error flag of third oscillator of first triple-oscillator cell

Bit 0 **TO1_REPET_ERROR**: Repetition error flag of third oscillator of first triple-oscillator cell.

14.7.18 TRNG Interrupt Control Register (TRNG_IRQ_CR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_ERROR_IRQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_FF_FULL_IRQ
r	r	r	r	r	r	r	rw	r	r	r	r	r	r	r	rw

Address: BaseAddress + 0x080

Type: R/W

Reset value: 0x0000 0000

Description: The TRNG_IRQ_CR register controls the enabling of interrupt sources.

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **EN_ERROR_IRQ**: Enable the interrupt when an error is reported.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **EN_FF_FULL_IRQ**: Enable the interrupt when the output FIFO is full of new random.

14.7.19 Interrupt Status Register (TRNG_IRQ_SR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERROR_IRQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FF_FULL_IRQ
r	r	r	r	r	r	r	rw	r	r	r	r	r	r	r	rw

Address: Base Address+ 0x084

Type: R

Reset value: 0x0000 0000

Description: The TRNG_IRQ_SR register reports the status flags of the interrupt sources.

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **ERROR_IRQ**: Set to 1 when an error is reported. Flag is cleared by writing a 1.
This bit is set only if corresponding bit in TRNG_IRQ_CR is set.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **FF_FULL_IRQ**: Set to 1 when the output FIFO is full of new random. It is software responsibility to clear the flag since it is not cleared by hardware when samples are read from the FIFO.
This bit is set only if corresponding bit in TRNG_IRQ_CR is set.

15 Cyclic redundancy check calculation unit (CRC)

15.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

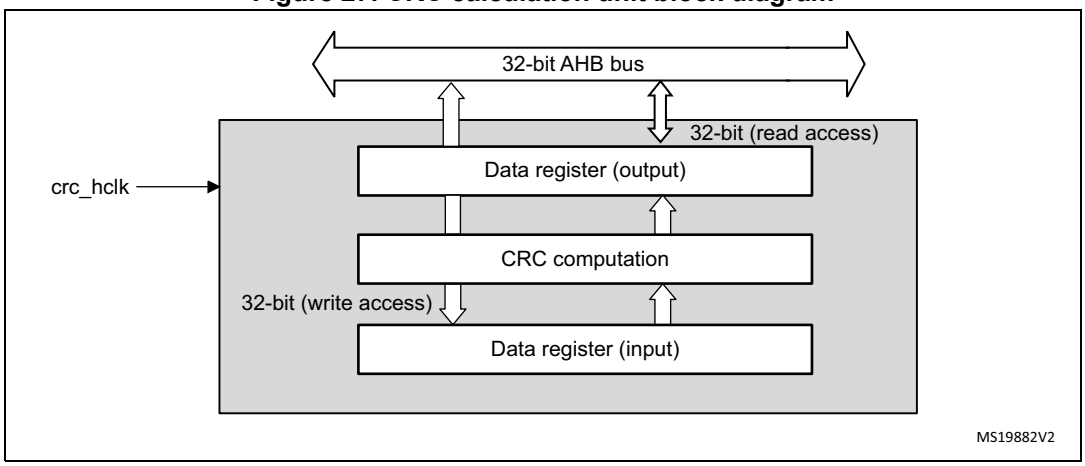
15.2 CRC main features

- Fully programmable polynomial with programmable size (7, 8, 16, 32 bits).
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data

15.3 CRC functional description

15.3.1 CRC block diagram

Figure 27. CRC calculation unit block diagram



15.3.2 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit access is allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycles for 8-bit

An input buffer allows to immediately write a second data without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example: input data 0x1A2B3C4D is used for CRC calculation as:

0x58D43CB2 with bit-reversal done by byte

0xD458B23C with bit-reversal done by half-word

0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

15.4 CRC registers

15.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.
It holds the previous CRC calculation result when it is read.
If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

15.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]:** This field can be used to hold a temporary value related to the CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

15.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rw

Bits 31:8 Reserved, must be kept cleared.

Bit 7 **REV_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data.

0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

These bits control the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept cleared.

Bit 0 **RESET**: Reset bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

15.4.4 Initial CRC value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]***:* Programmable initial CRC value. This register is used to write the CRC initial value.

15.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C11DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: CRC polynomial coefficients. This allows programming of the polynomial coefficients.

15.4.6 CRC register map

Table 78. PKA register map

Offset	Register	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
0x00	**CRC_DR**	DR[31:0]																															
	Reset value	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
0x04	**CRC_IDR**	IDR[31:0]																															
	Reset value	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
0x08	**CRC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0					0
0x10	**CRC_INIT**	CRC_INIT[31:0]																															
	Reset value	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
0x14	**CRC_POL**	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

16 General purpose timer (TIM2)

In this section, “TIMx” refers to TIM2, since there is only one instance of this timer in the STM32WB09xE device.

16.1 TIM2 introduction

The general purpose timers (TIM2) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

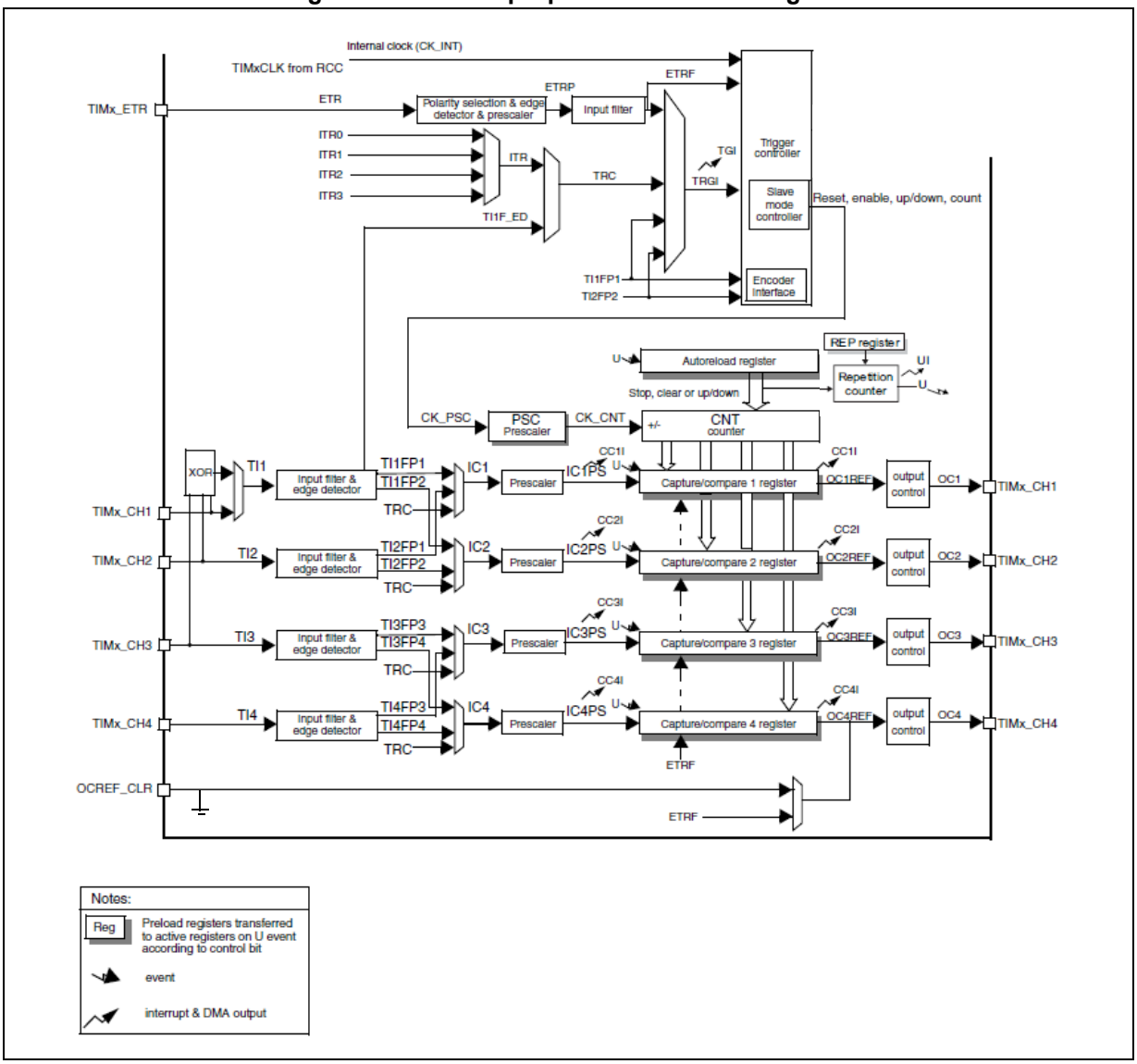
Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler on the timer input clock which is at 32 MHz.

16.2 TIM2 main features

TIM2 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input Capture
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger) (only interrupt)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 28. General purpose timer block diagram



16.3 TIM2 functional description

16.3.1 Time-base unit

The main block of the programmable general purpose timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 29 and *Figure 30* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 29. Counter timing diagram with prescaler division change from 1 to 2

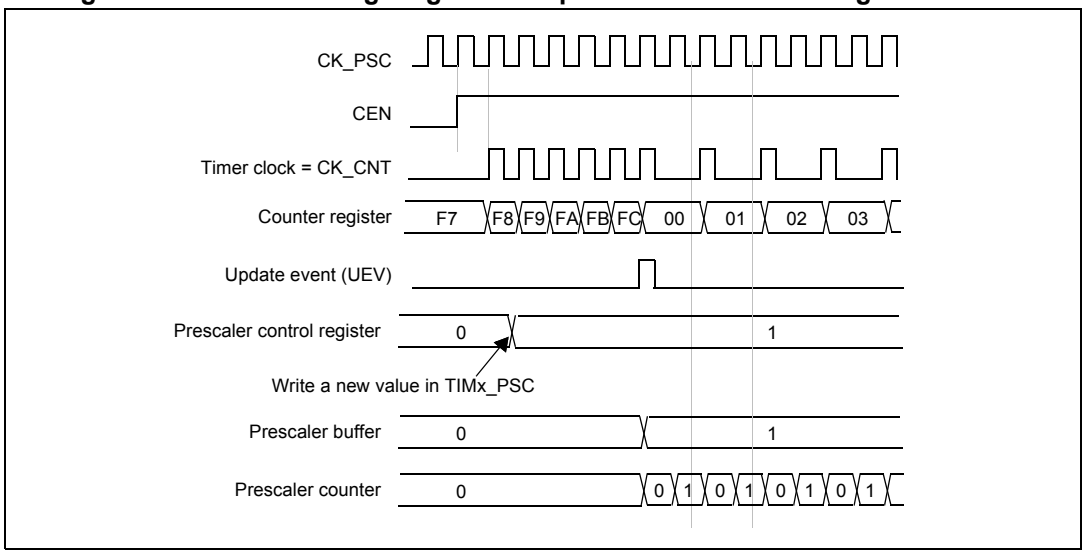
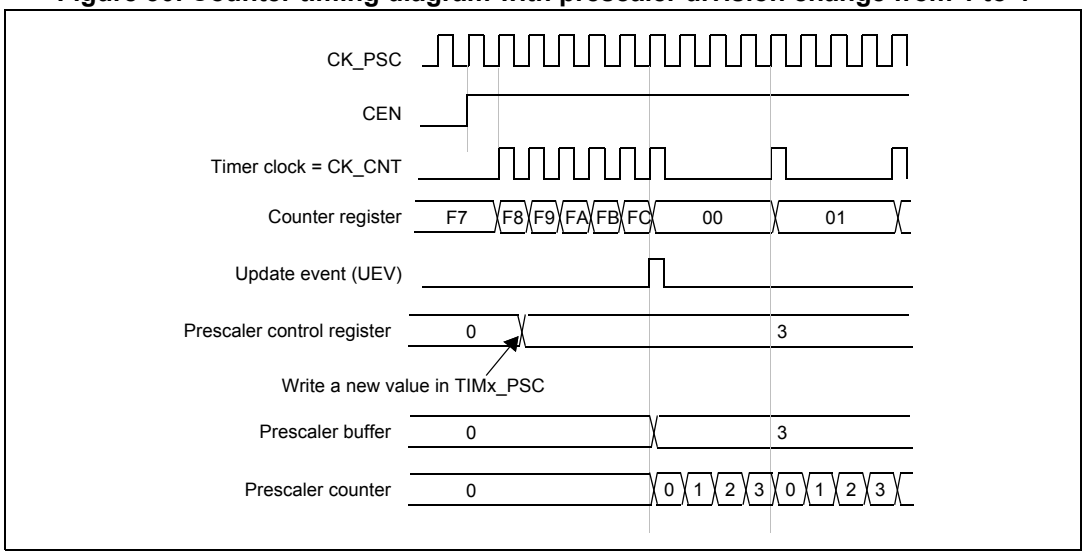


Figure 30. Counter timing diagram with prescaler division change from 1 to 4



16.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 31. Counter timing diagram, internal clock divided by 1

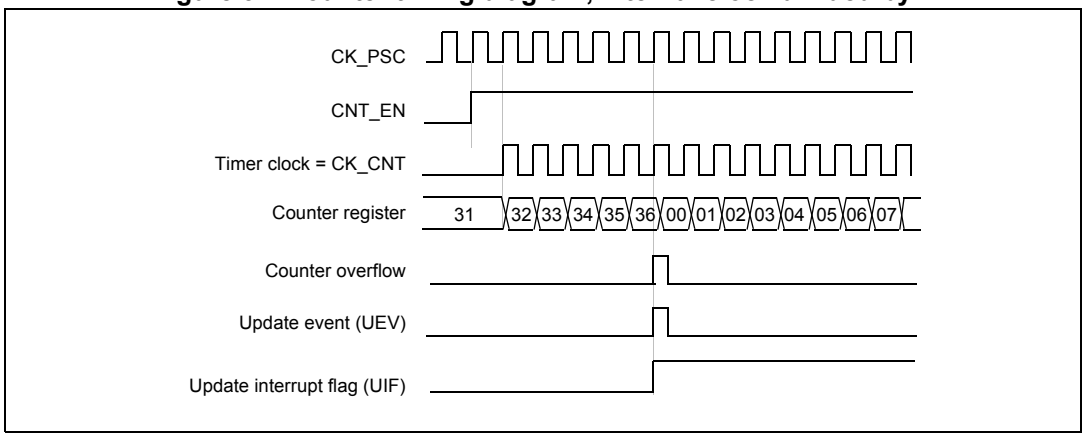


Figure 32. Counter timing diagram, internal clock divided by 2

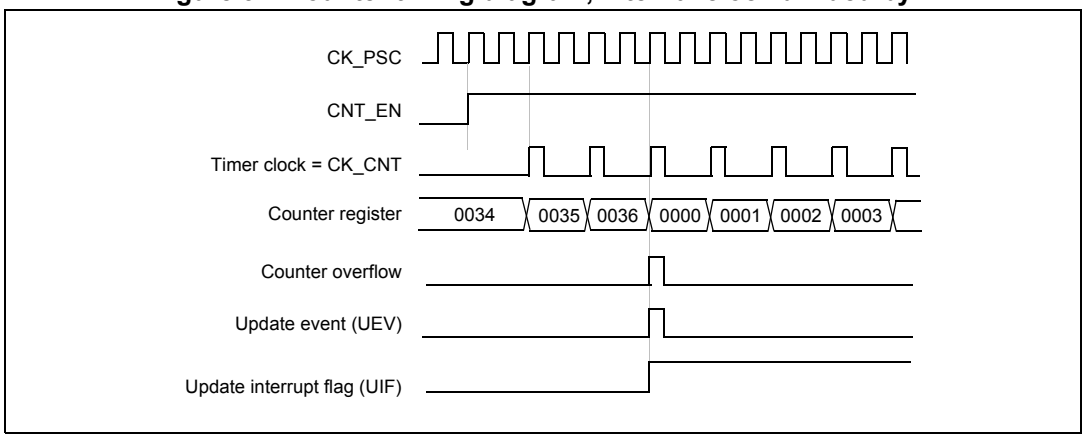


Figure 33. Counter timing diagram, internal clock divided by 4

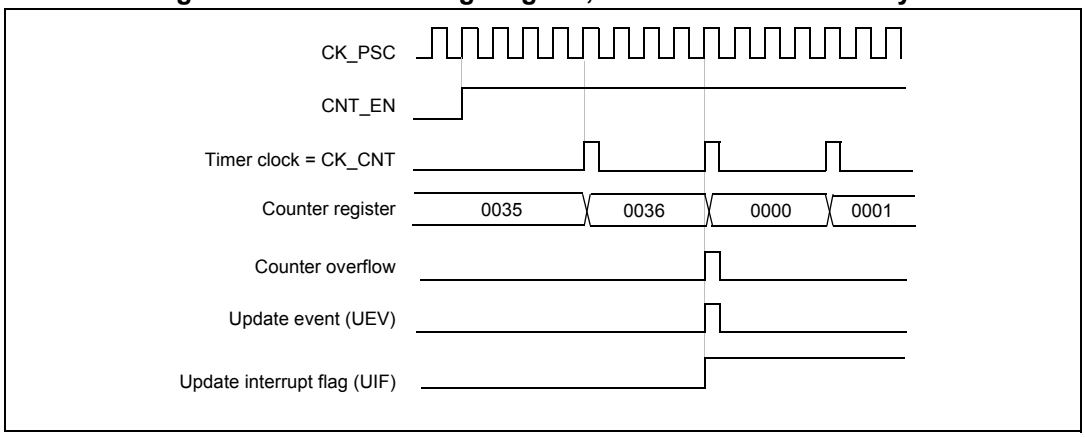


Figure 34. Counter timing diagram, internal clock divided by N

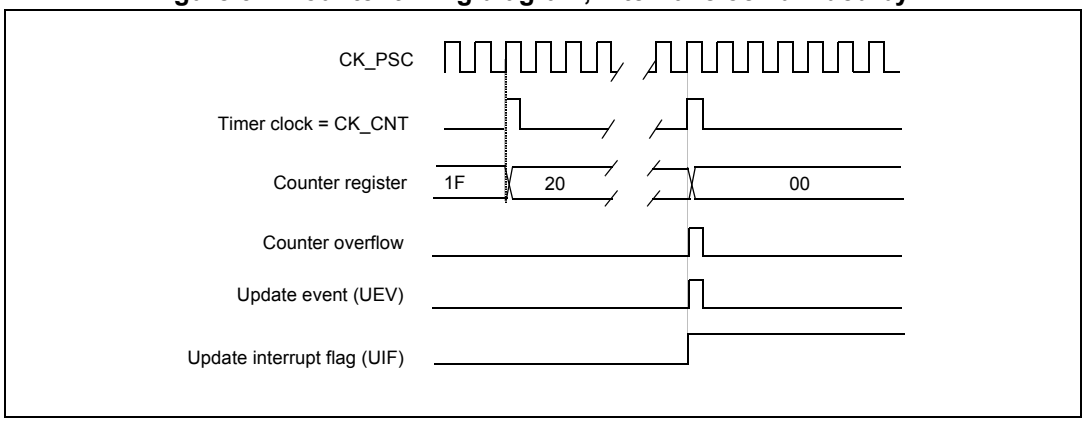


Figure 35. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

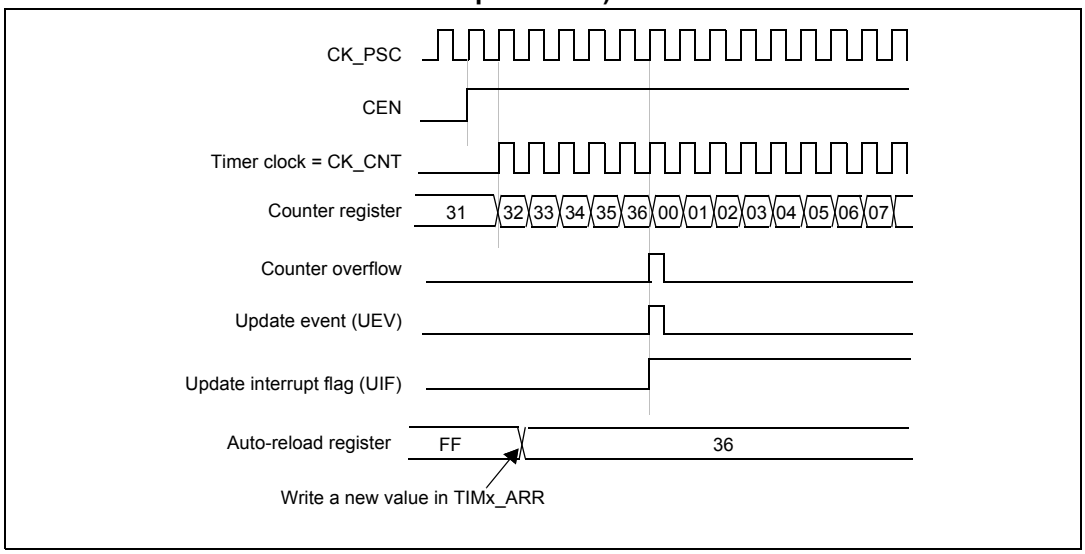
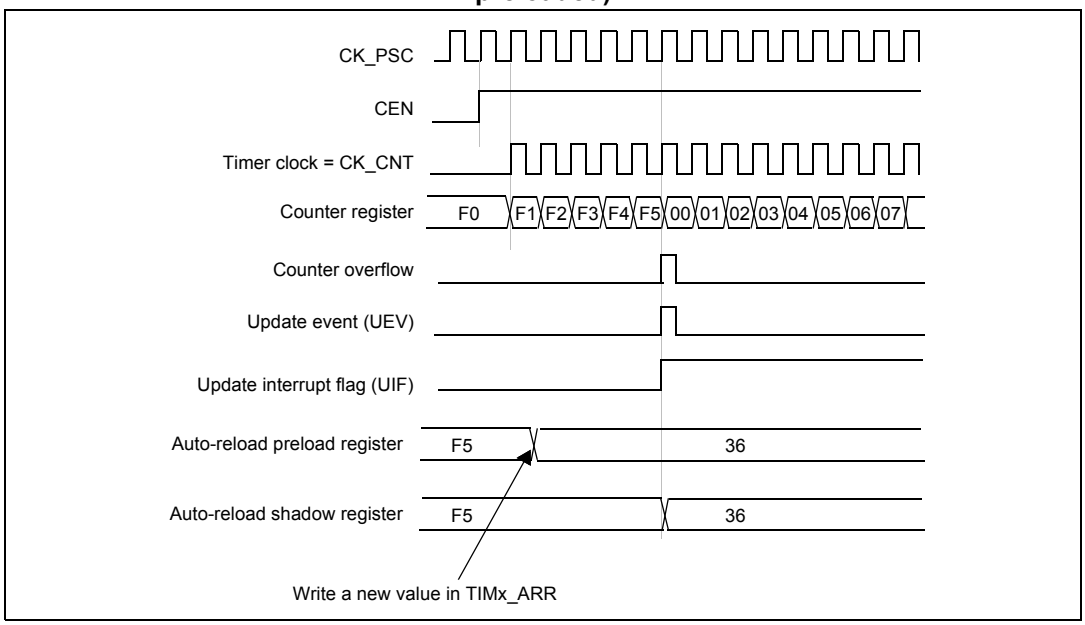


Figure 36. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 37. Counter timing diagram, internal clock divided by 1

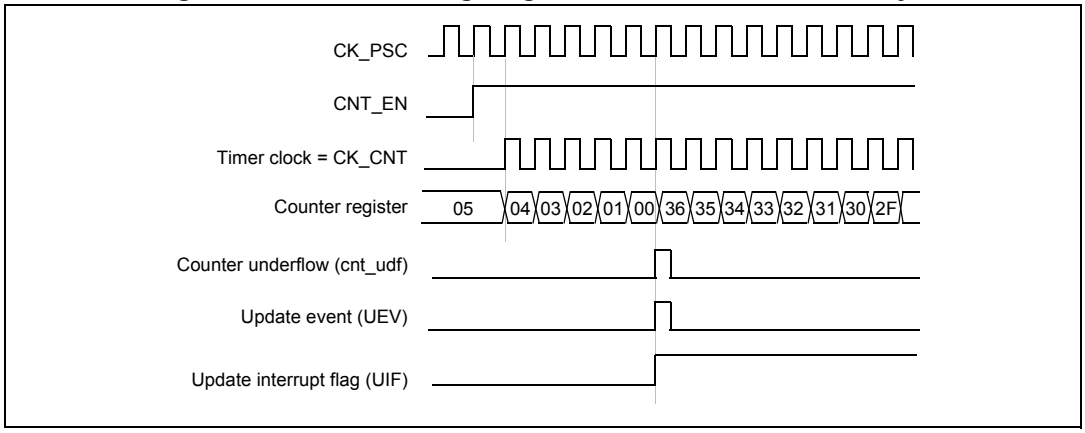


Figure 38. Counter timing diagram, internal clock divided by 2

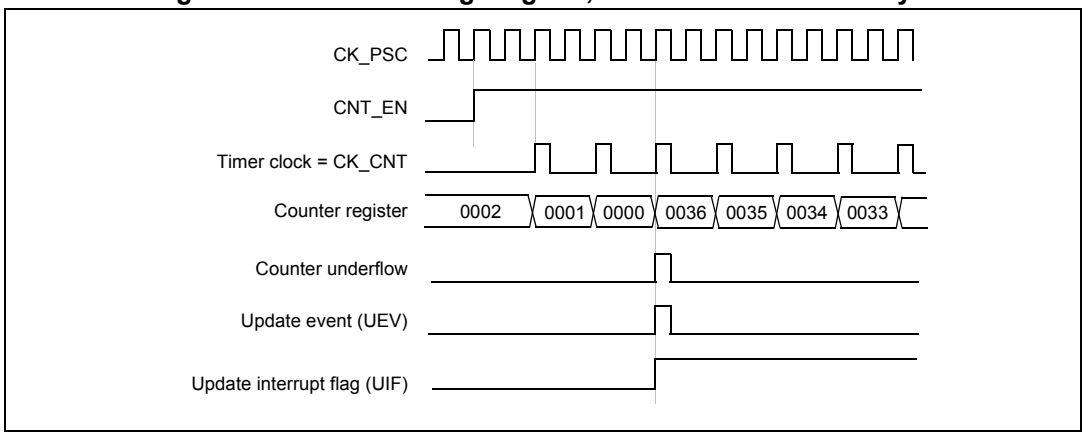


Figure 39. Counter timing diagram, internal clock divided by 4

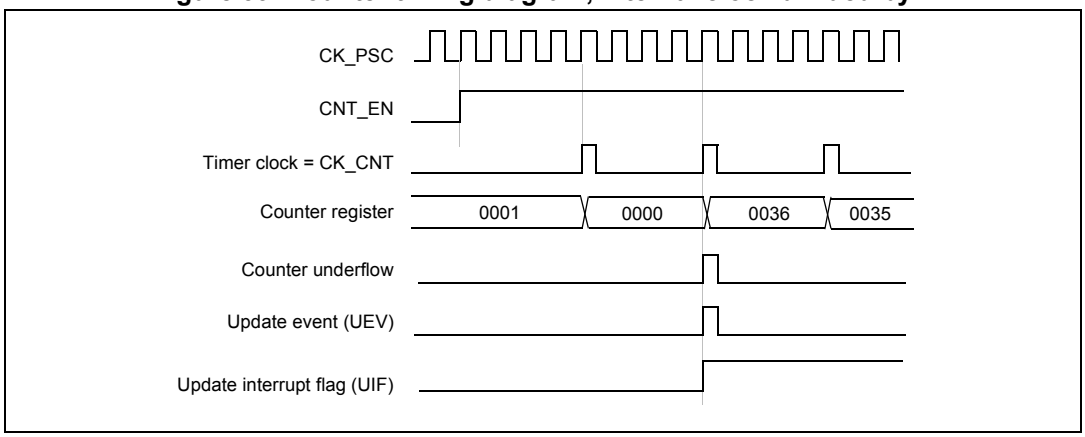


Figure 40. Counter timing diagram, internal clock divided by N

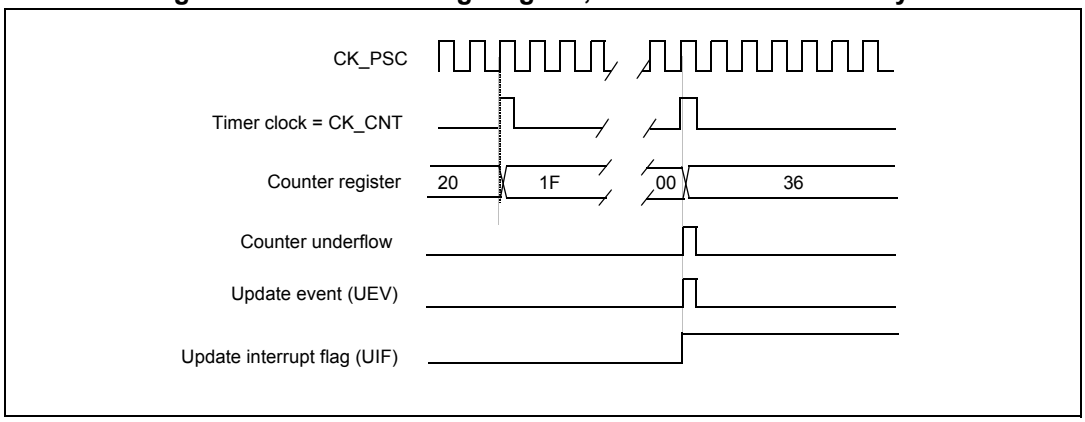
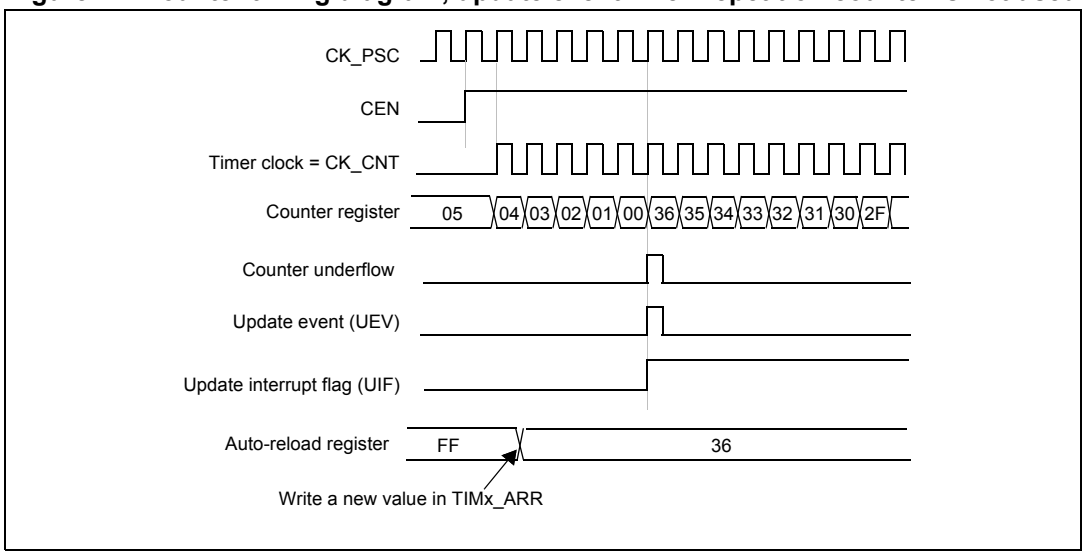


Figure 41. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

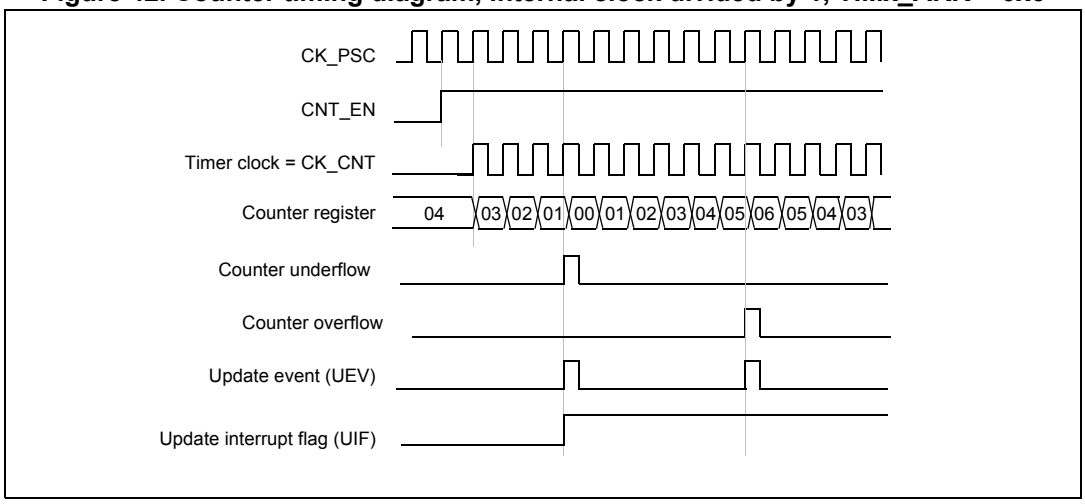
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 42. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 16.4: TIM2 registers*).

Figure 43. Counter timing diagram, internal clock divided by 2

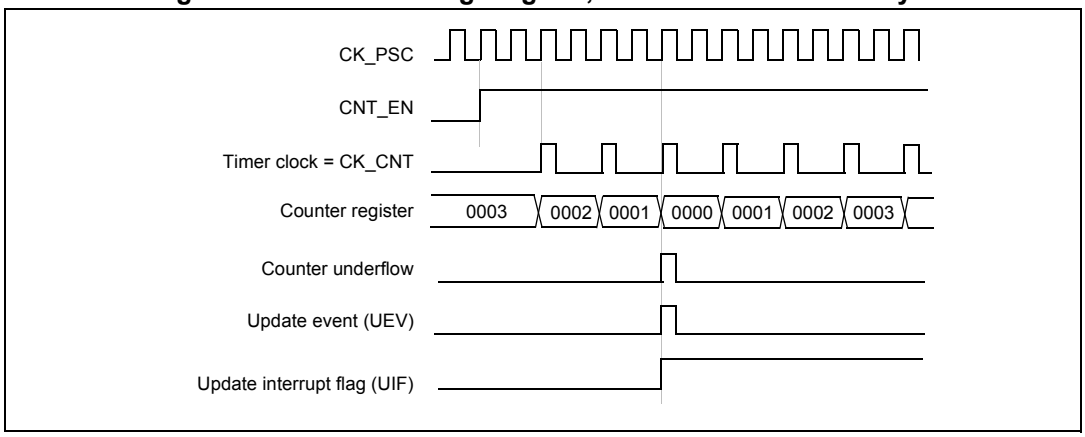


Figure 44. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

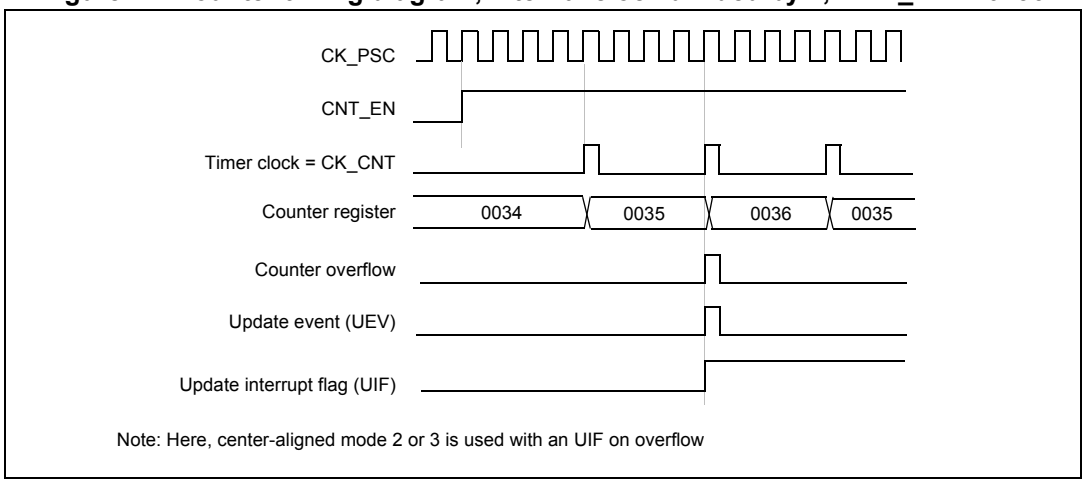


Figure 45. Counter timing diagram, internal clock divided by N

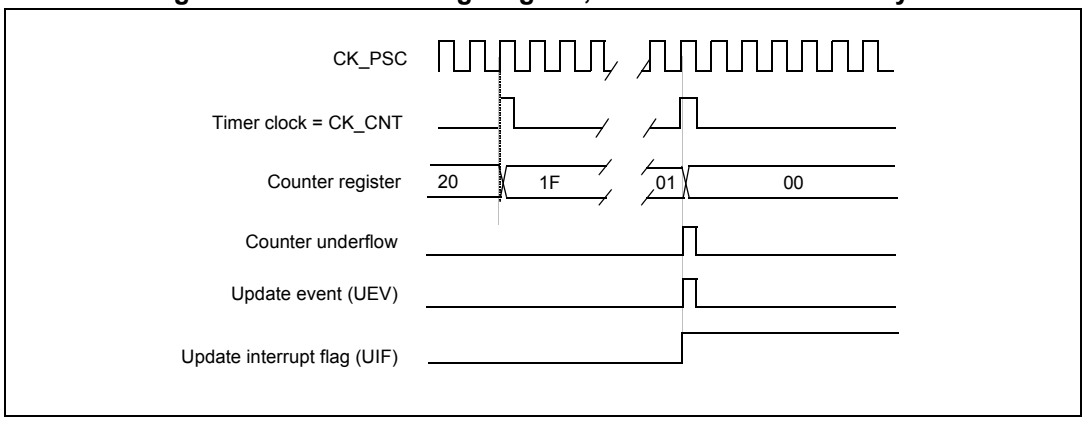


Figure 46. Counter timing diagram, update event with ARPE=1 (counter underflow)

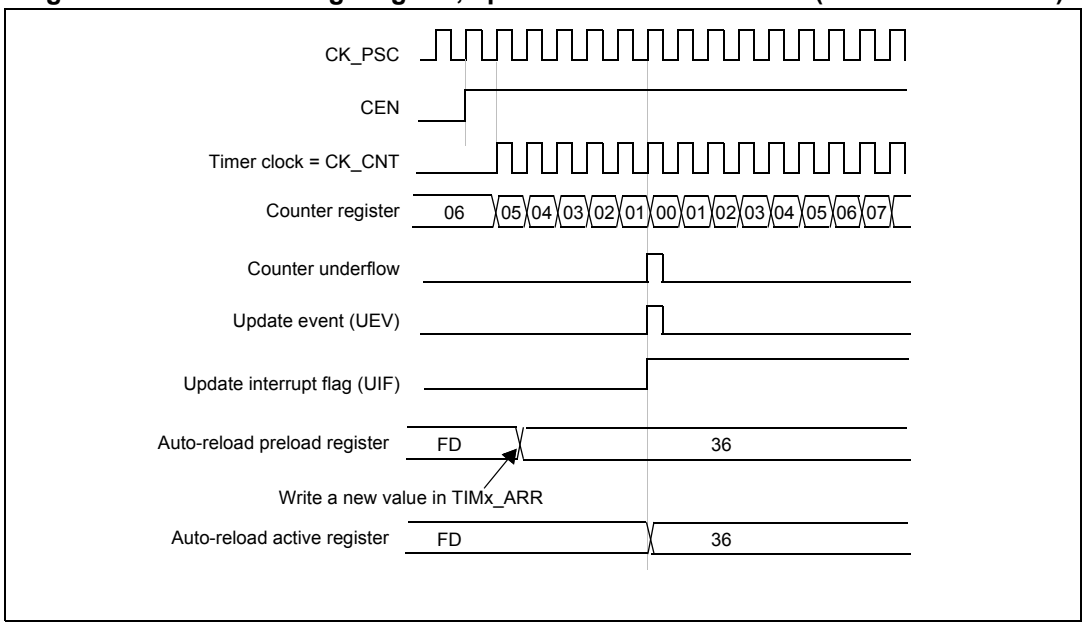
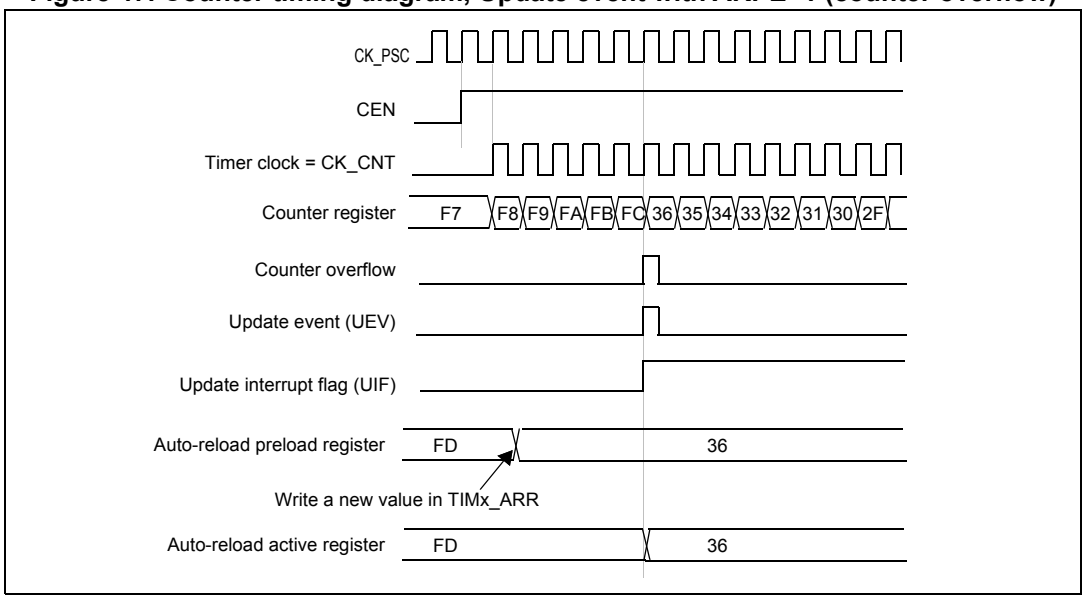


Figure 47. Counter timing diagram, Update event with ARPE=1 (counter overflow)



16.3.3 Repetition counter

Section 16.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

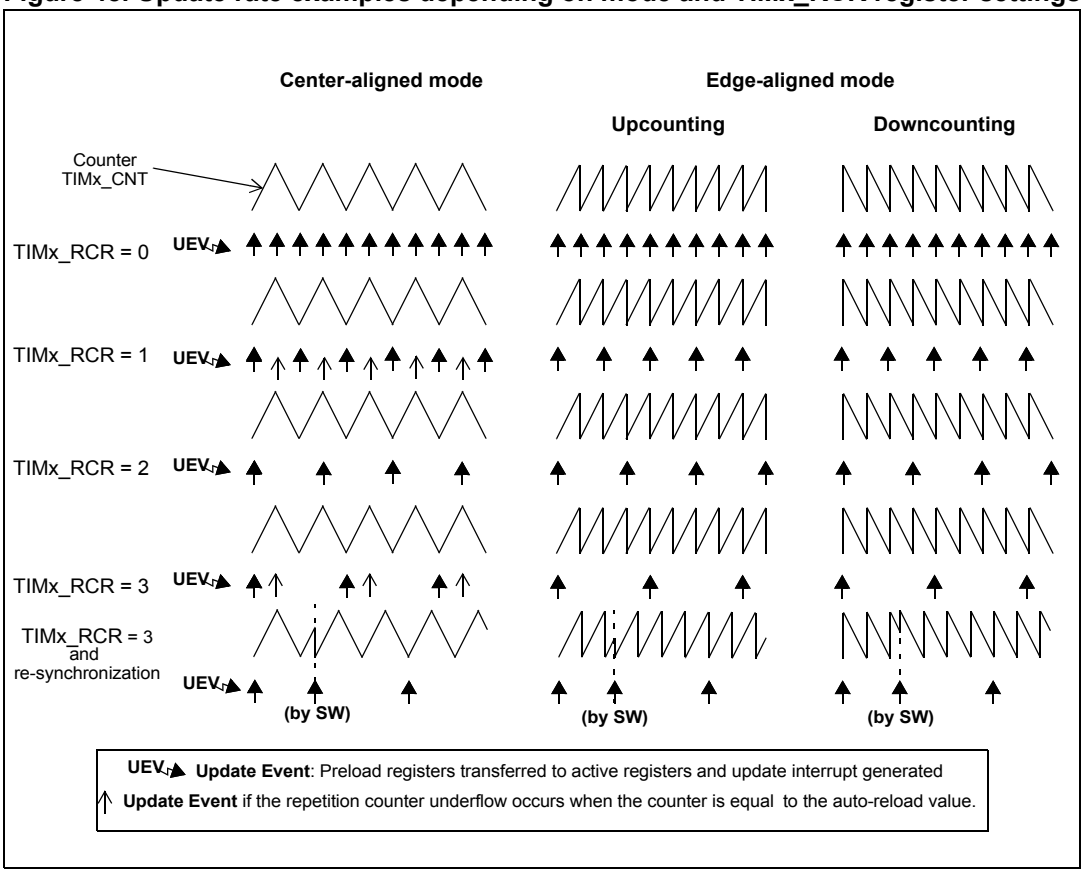
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 48*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the overflow. If the RCR was written after launching the counter, the UEV occurs on the underflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 48. Update rate examples depending on mode and TIMx_RCR register settings



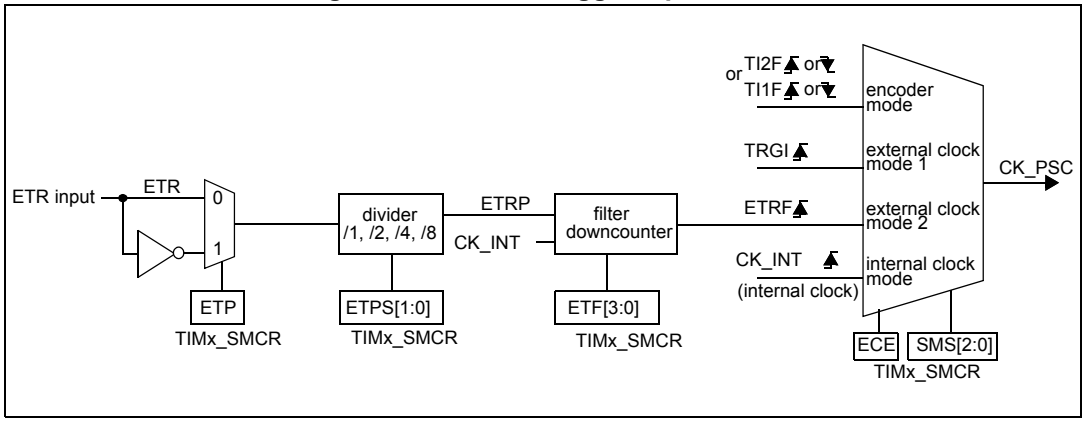
16.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see *Section 16.3.5*)
- trigger for the slave mode (see *Section 16.3.4*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 16.3.14*)

The *Figure 49* below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMx_SMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bit field.

Figure 49. External trigger input block



The ETR input comes from input pins (see *Table 7: Alternate modes 0, 1 and 2* and *Table 8: Alternate modes 3, 4, and 6*).

16.3.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin

Note: Only channel 1 and channel 2 support the external clock mode 1.

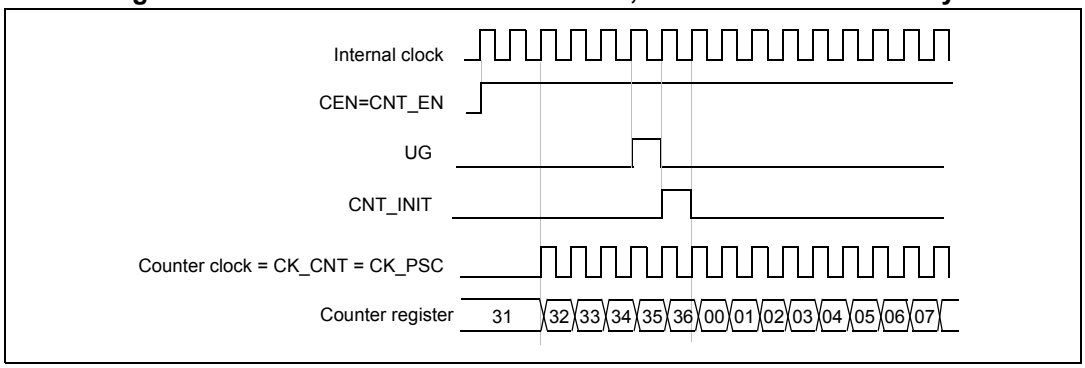
- External clock mode2: external trigger input ETR
- Encoder mode

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 50 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

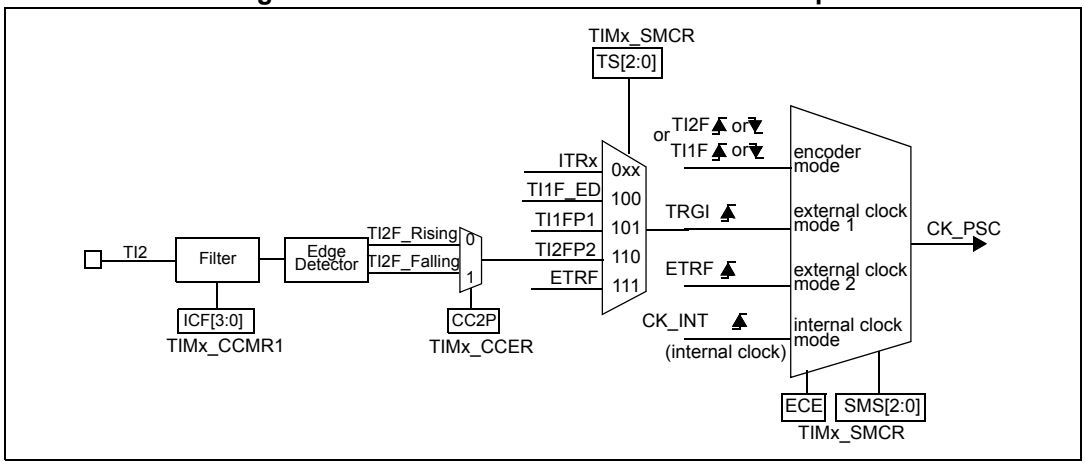
Figure 50. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 51. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

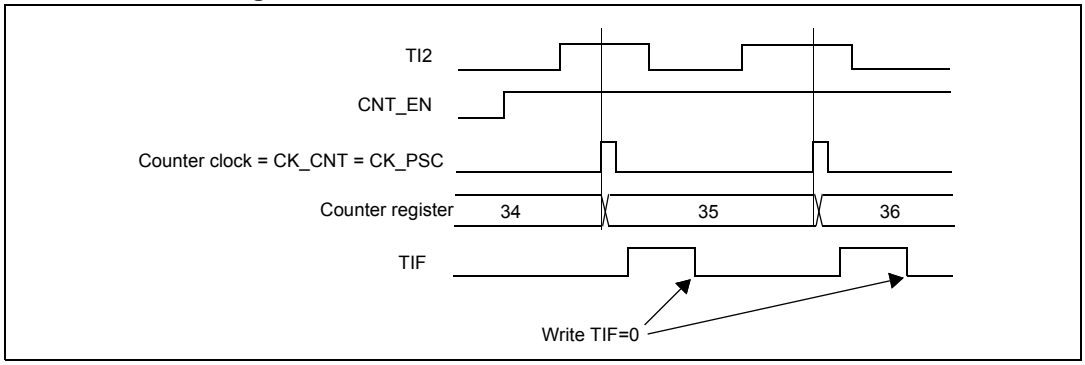
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don’t need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 52. Control circuit in external clock mode 1



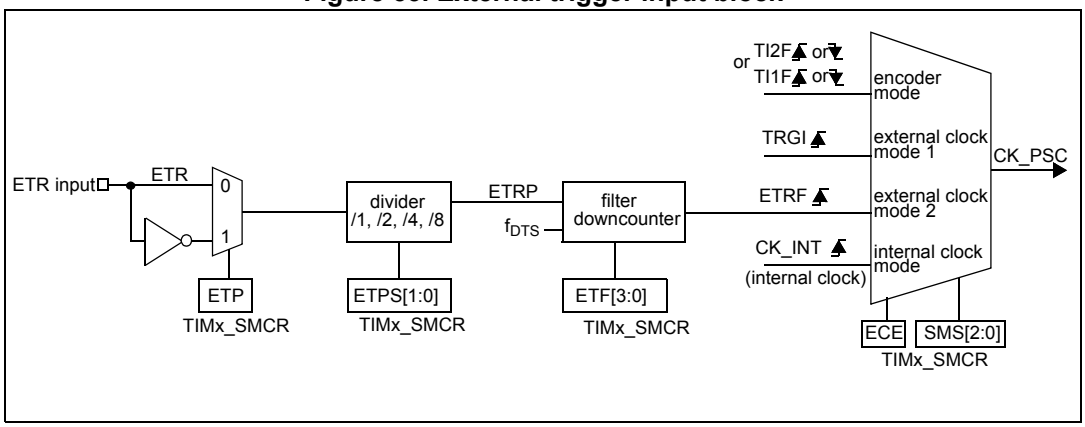
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 53* gives an overview of the external trigger input block.

Figure 53. External trigger input block



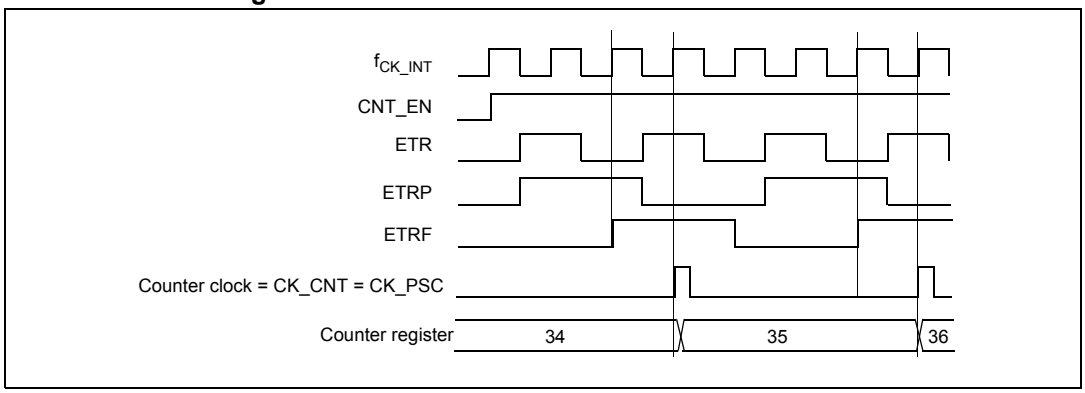
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 54. Control circuit in external clock mode 2



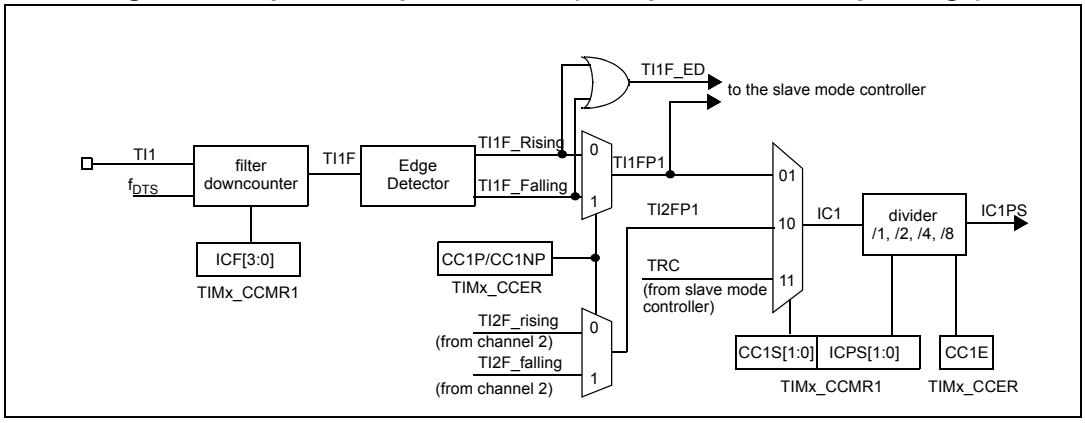
16.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 55 to *Figure 57* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 55. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 56. Capture/compare channel 1 main circuit

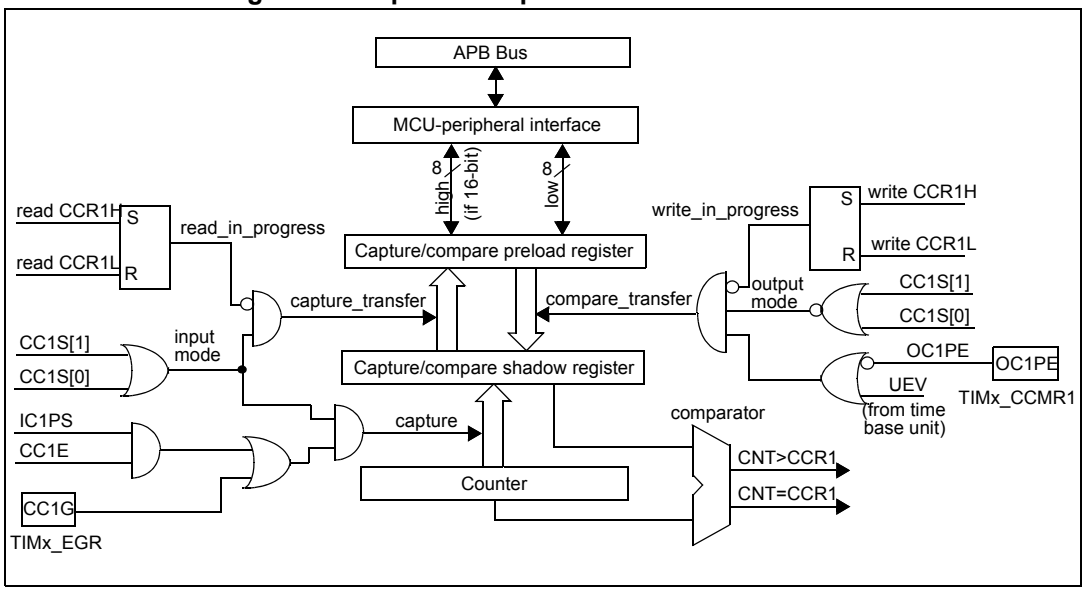
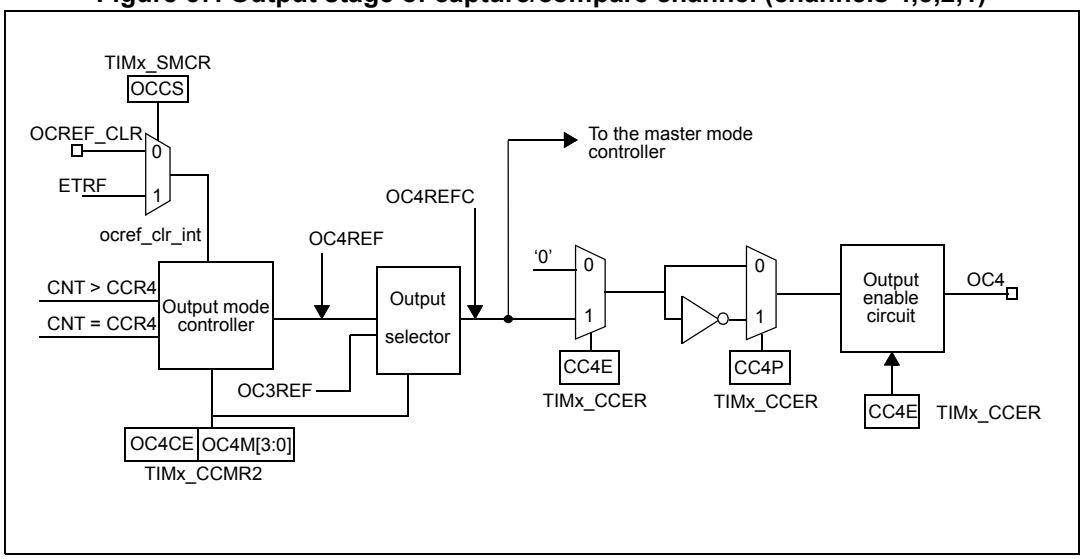


Figure 57. Output stage of capture/compare channel (channels 4,3,2,1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

16.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

16.3.8 PWM input mode

Note: Only channel 1 and channel 2 support this PWM input mode.

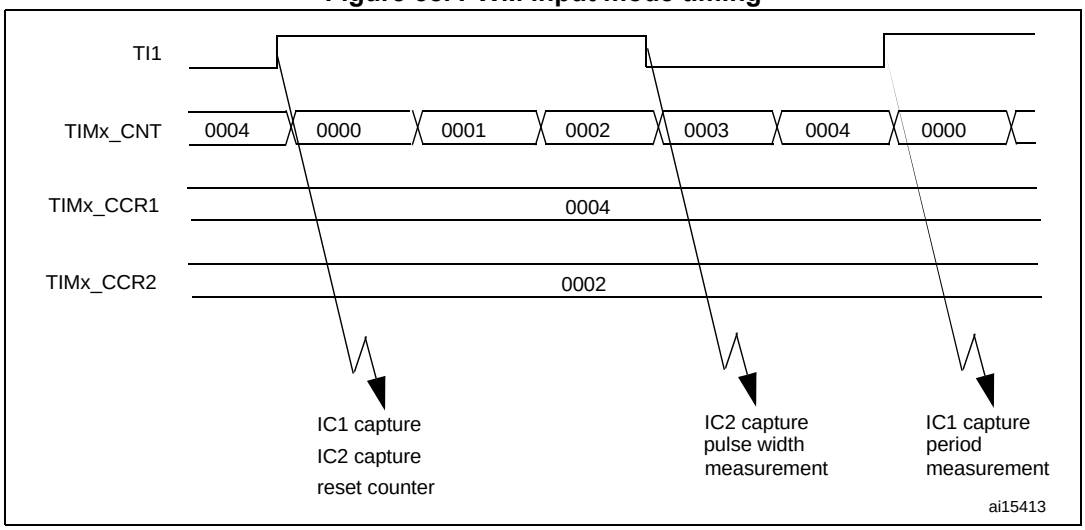
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 58. PWM input mode timing



16.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

16.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 6 can be output.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

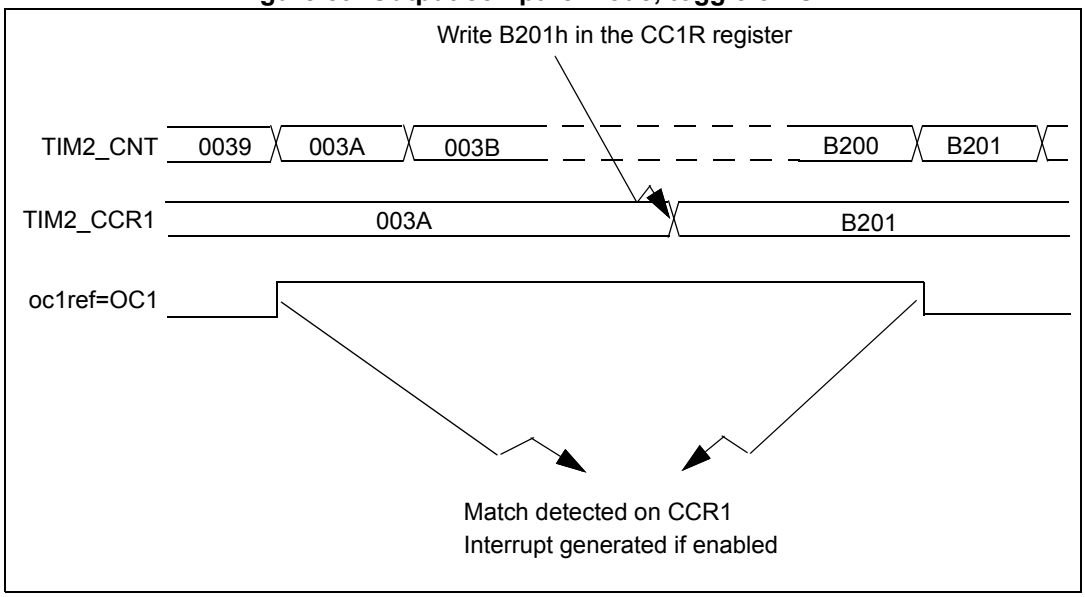
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 59*.

Figure 59. Output compare mode, toggle on OC1



16.3.11 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤TIMx_CNT or TIMx_CNT ≤TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

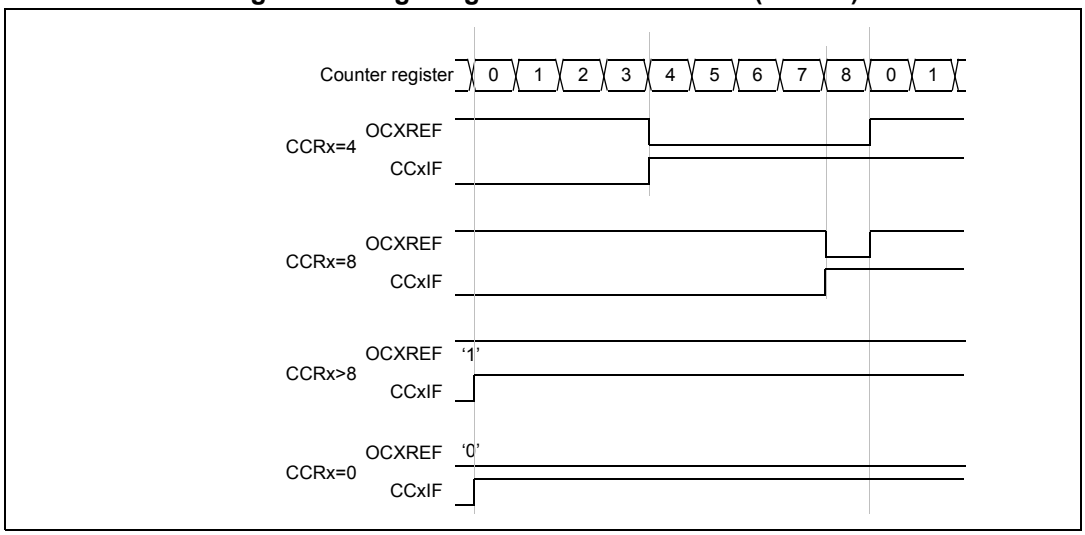
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the Up *Counter modes on page 310*.

  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 60* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 60. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode on page 313*

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at '1'. 0% PWM is not possible in this mode.

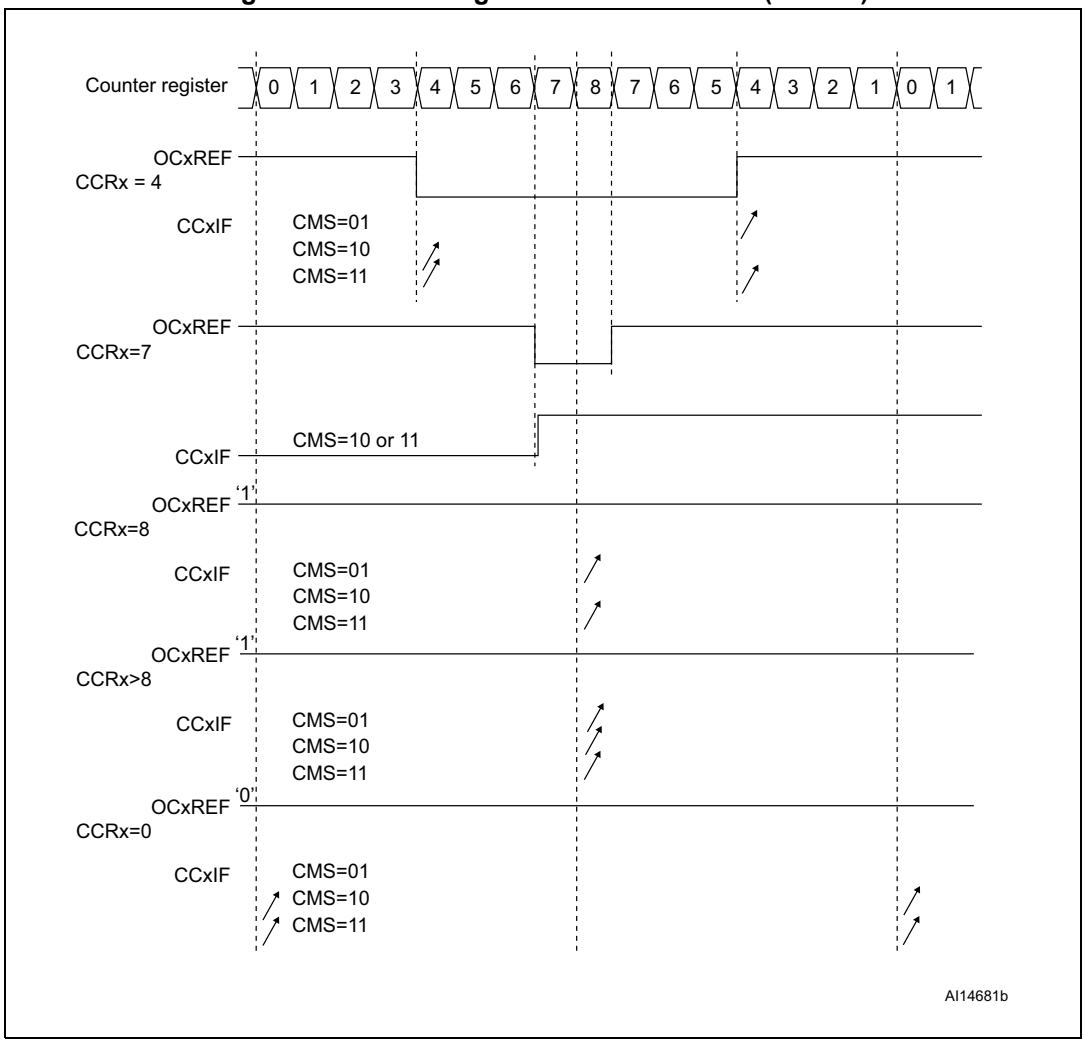
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting) on page 315*.

Figure 61 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 61. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

16.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

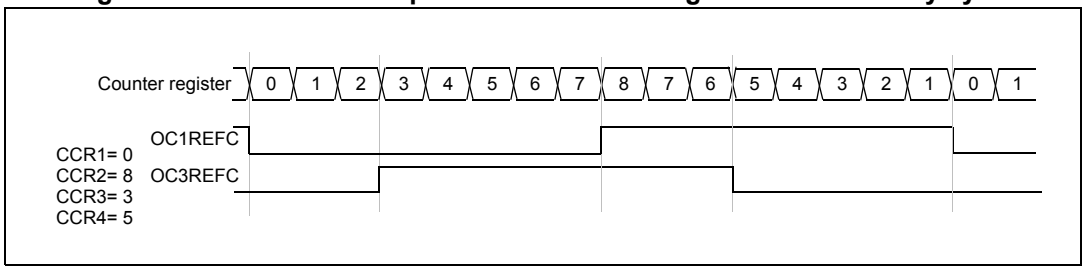
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

Figure 62 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 62. Generation of 2 phase-shifted PWM signals with 50% duty cycle



16.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

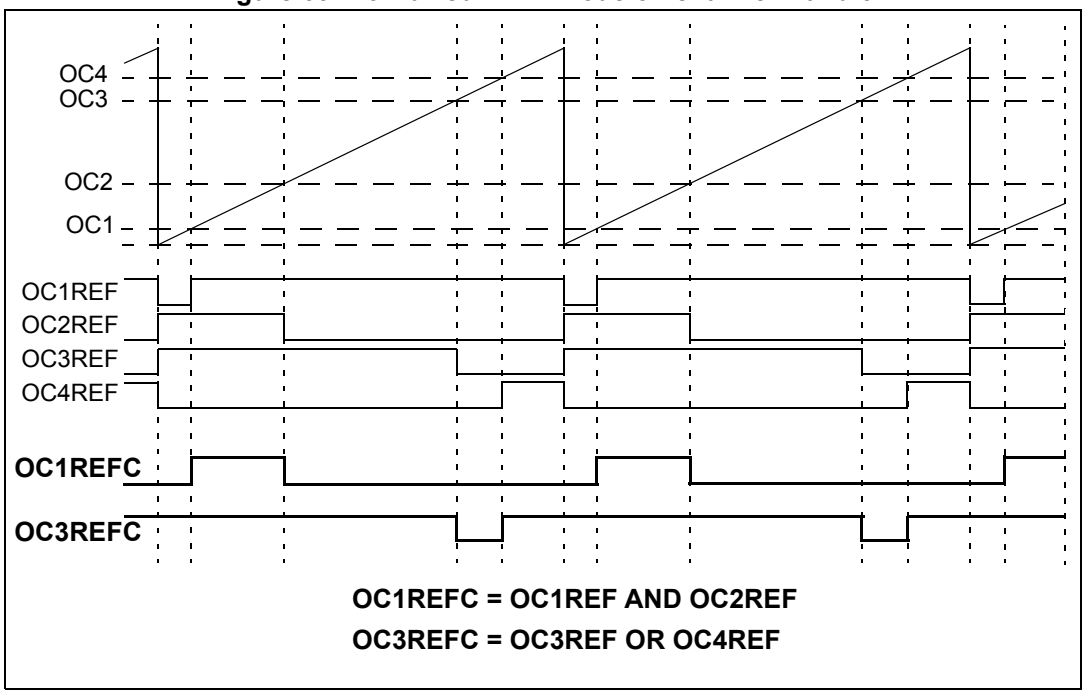
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 63 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.

Figure 63. Combined PWM mode on channel 1 and 3



16.3.14 Clearing the OCxREF signal on an external event

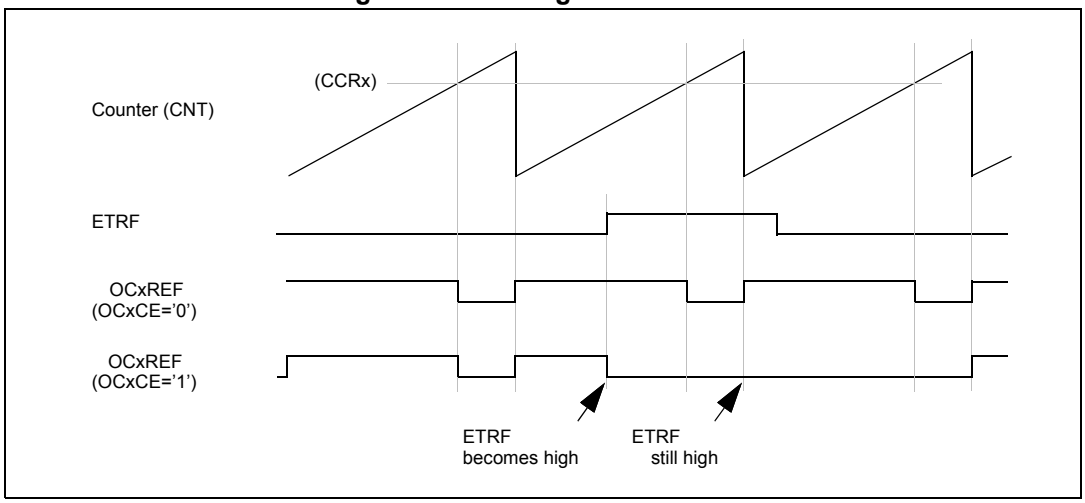
The OCxREF signal of a given channel can be cleared when a high level is applied on the ETRF input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1) if TIMx_SMCR.OCCS bit is set to 1. The OCxREF signal remains low until the next update event (UEV) occurs.

This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 64 shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 64. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.

16.3.15 One-pulse mode

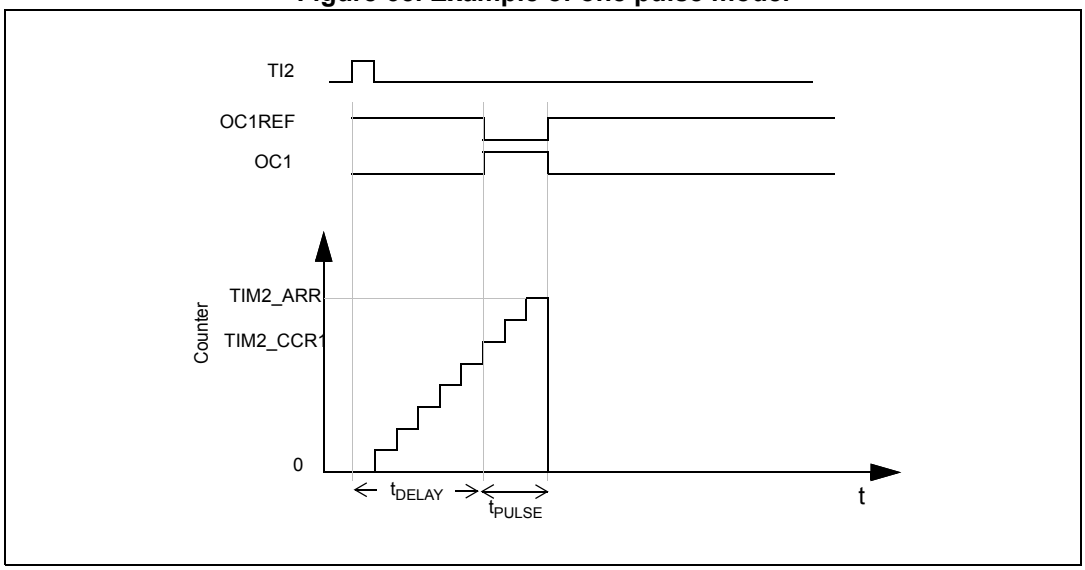
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 65. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

16.3.16 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 16.3.15*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

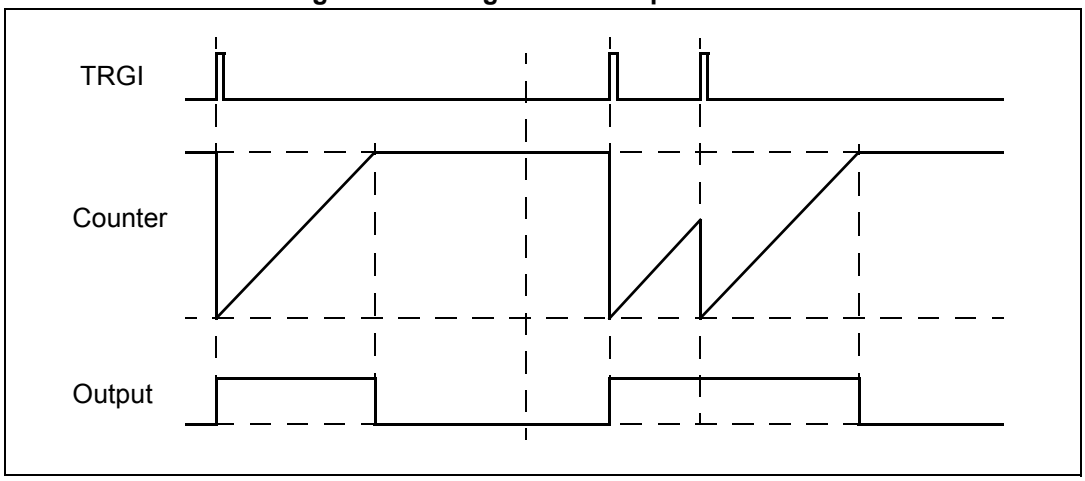
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, the ARR must be set to 0 (the CCRx register sets the pulse length).

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

In Retriggerable one pulse mode, the CCxIF flags are not significant.

Figure 66. Retrigerrable one pulse mode



16.3.17 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, you can program the input filter as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an quadrature encoder. Refer to *Table 79*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must configure TIMx_ARR before starting. in the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don't switch at the same time.

Table 79. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The *Figure 67* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 67. Example of counter operation in encoder interface mode.

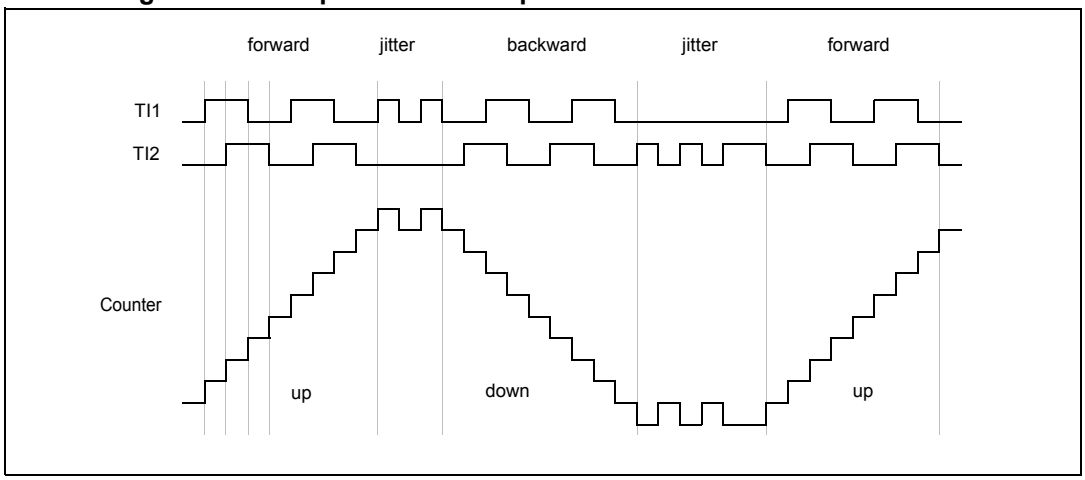
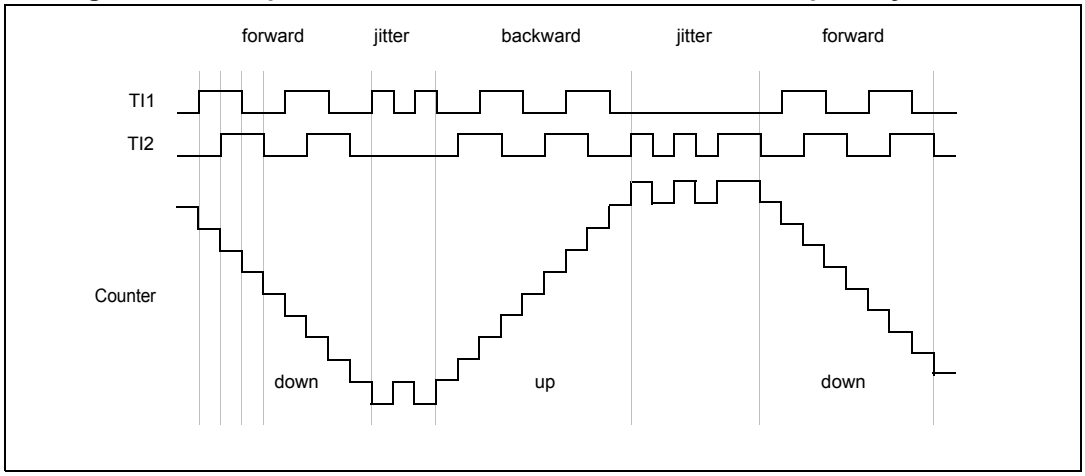


Figure 68 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 68. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode (not available in the STM32WB09xE device). The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer), when available it's also possible to read its value through a DMA request generated by a Real -Time clock.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

16.3.18 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

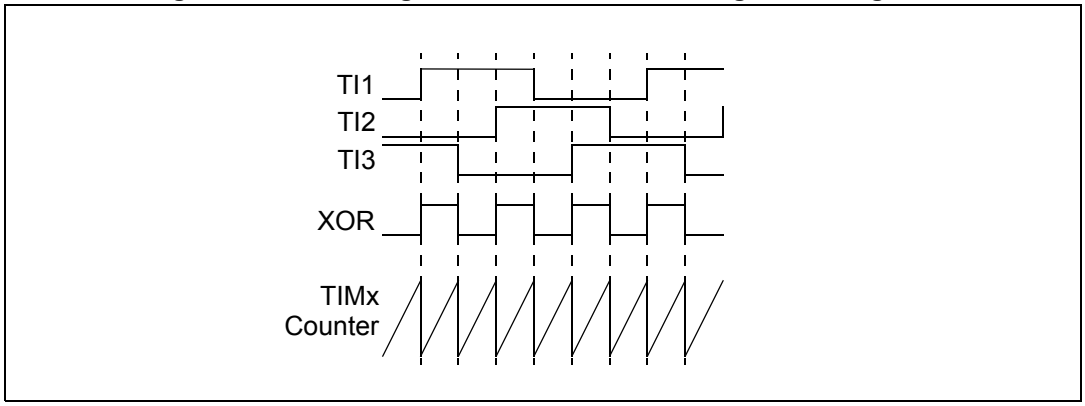
There is no latency between the UIF and UIFCPY flags assertion.

16.3.19 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 69* below.

Figure 69. Measuring time interval between edges on 3 signals



16.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

16.4 TIM2 registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

16.4.1 TIM2 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, always read as 0

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10 Reserved, always read as 0

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx),
00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

16.4.2 TIM2 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
								rw	rw	rw	rw		rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS Res.	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	Res.	Res.	Res.	CCDS	Res.	Res.	Res.
								rw							

Bits 31:8 Reserved, always read as 0

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:
0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.
0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).
0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.
0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).
0100: **Compare** - OC1REF signal is used as trigger output (TRGO2)
0101: **Compare** - OC2REF signal is used as trigger output (TRGO2)
0110: **Compare** - OC3REF signal is used as trigger output (TRGO2)
0111: **Compare** - OC4REF signal is used as trigger output (TRGO2)
1000: **Compare** - OC5REF signal is used as trigger output (TRGO2)
1001: **Compare** - OC6REF signal is used as trigger output (TRGO2)
1010: **Compare Pulse** - OC4REF rising or falling edges generate pulses on TRGO2
1011: **Compare Pulse** - OC6REF rising or falling edges generate pulses on TRGO2
1100: **Compare Pulse** - OC4REF or OC6REF rising edges generate pulses on TRGO2
1101: **Compare Pulse** - OC4REF rising or OC6REF falling edges generate pulses on TRGO2
1110: **Compare Pulse** - OC5REF or OC6REF rising edges generate pulses on TRGO2
1111: **Compare Pulse** - OC5REF rising or OC6REF falling edges generate pulses on TRGO2

Bit 19 Reserved, always read as 0

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[1:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REF signal is used as trigger output (TRGO)
101: **Compare** - OC2REF signal is used as trigger output (TRGO)
110: **Compare** - OC3REF signal is used as trigger output (TRGO)
111: **Compare** - OC4REF signal is used as trigger output (TRGO)

Bits 6:4 Reserved, always read as 0.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2:0 Reserved, always read as 0

16.4.3 TIM2 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				Res.	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, always read as 0

Bits 19:17 Reserved, always read as 0

Bit 16 **SMS[3]**: Slave mode selection - bit 3
Refer to SMS description - bits2:0

Bit 15 **ETP**: External trigger polarity
This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: **1:** Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).*
***2:** It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).*
***3:** If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler
External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.
00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter
This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 Reserved, always read as 0

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)

00001: Internal Trigger 1 (ITR1)

00010: Internal Trigger 2 (ITR2)

00011: Internal Trigger 3 (ITR3)

00100: TI1 Edge Detector (TI1F_ED)

00101: Filtered Timer Input 1 (TI1FP1)

00110: Filtered Timer Input 2 (TI2FP2)

00111: External Trigger input (ETRF)

Others: Reserved

See *Table Note:: TIM2 internal trigger connection on page 348* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: OCREF_CLR_INT is connected to the OCREF_CLR input (stuck at 0 so no effect)

1: OCREF_CLR_INT is connected to ETRF

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Others: Reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.The clock of the slave peripherals (timer2) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: TIM2 internal trigger connection

Slave TIM	ITR0	ITR1	ITR2 - ITR8
TIM2	TIM16	-	-

Table 80. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	n.a.	TIM2	n.a.	TIM17 OC1

16.4.4 TIM2 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
			rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 15:13 Reserved, always read as 0.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 15 Reserved, always read as 0.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 Reserved, always read as 0.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 Reserved, always read as 0.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

16.4.5 TIM2 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:13 Reserved, always read as 0.

Bit 13 **SBIF**: System Break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8:7 Reserved, always read as 0.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, always read as 0.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag
If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 16.4.3: TIM2 slave mode control register (TIMx_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

16.4.6 TIM2 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, always read as 0.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt can occur if enabled.

Bits 5 Reserved, always read as 0.

Bit 4 **CC4G**: Capture/Compare 4 generation

refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

16.4.7 TIM2 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							Res.								Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bits16 **OC1M[3]**: Output Compare 1 mode - bit 3
Refer to OC1M description on bits 6:4

Bit 15 **OC2CE:** Output Compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output Compare 1 clear enable
OC1CE: Output Compare 1 Clear Enable
0: OC1Ref is not affected by the ETRF Input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0’) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=’1’).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).*

***2:** In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

16.4.8 TIM2 capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							Res.								Res.
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC4M[3]**: Output Compare 4 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bit 16 **OC3M[3]**: Output Compare 3 mode - bit 3

Bit 15 OC4CE: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

16.4.9 TIM2 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bits 31:16 Reserved, always read as 0.

Bit 15 **CC4NP**: Capture/compare 4 output complementary polarity. Refer to CC1P description.

Bit 14 Reserved, always read as 0.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity. Refer to CC1P description.

Bit 10 Reserved, always read as 0.

Bit 9 **CC3P**: Capture/Compare 3 output polarity
refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity. Refer to CC1P description.

Bit 6 Reserved, always read as 0.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity. Refer to CC1P description.

Bit2 Reserved, always read as 0.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
00: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
01: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
10: reserved, do not use this configuration.
11: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: On channels having a complementary output, this bit is preloaded.

Bit 0 **CC1E**: Capture/Compare 1 output enable
CC1 channel configured as output:
0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Note: On channels having a complementary output, this bit is preloaded.

Table 81. Output control bits for OCx channels

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z) OCx=0
1	OCxREF + Polarity OCx=OCxREF xor CCxP

16.4.10 TIM2 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, always read as 0.

Bits 15:0 **CNT[15:0]**: Counter value

16.4.11 TIM2 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

16.4.12 TIM2 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 16.3.1: Time-base unit on page 309* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

16.4.13 TIM2 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								REP[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

the number of PWM periods in edge-aligned mode

the number of half PWM period in center-aligned mode.

16.4.14 TIM2 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

16.4.15 TIM2 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2).

16.4.16 TIM2 capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output:
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input:
CCR3 is the counter value transferred by the last input capture 3 event (IC3).

16.4.17 TIM2 capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output:
CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC4 output.
If channel CC4 is configured as input:
CCR4 is the counter value transferred by the last input capture 4 event (IC4).

16.4.18 TIM2 DMA control register (TIM2_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBL[4:0]					Res	Res	Res	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length
This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM2_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address
This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIM2_DMAR address). DBA is defined as an offset starting from the address of the TIM2_CR1 register.
Example:
00000: TIM2_CR1,
00001: TIM2_CR2,
00010: Reserved,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIM2_CR1. In this case the transfer is done to/from 7 registers starting from the TIM2_CR1 address.

16.4.19 TIM2 DMA address for full transfer (TIM2_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM2_CR1 address) + (DBA + DMA index) x 4

where TIM2_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM2_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM2_DCR).

16.4.20 TIM2 input selection register (TIM2_TISEL)

Address offset: 0x68

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input
0000: TIMx_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input
0000: TIMx_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input
0000: TIMx_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIMx_CH1 input
Others: Reserved

16.4.21 TIM2 register map

TIM2 registers are mapped as 16-bit addressable registers as described in the table below:

Table 82. TIM2 register map and reset values

Offset	Register	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
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Res	Res	Res	Res	Res	Res	Res	Res	MMS2[3:0]				Res	OIS6	Res	OIS5	Res	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x04	TIMx_CR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1S	Res	Res	Res	CCDS	Res	Res	Res
	Reset value																									0				0			
0x08	TIMx_SMCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	TIMx_SMCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				Res	TS[2:0]			OCCS	SMS[2:0]		
	Reset value																0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0C	TIMx_DIER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																										0		0	0	0	0	0
0x10	TIMx_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC4OF	CC3OF	CC2OF	CC1OF	Res	Res	TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	TIMx_EGR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TG	Res	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	TIMx_CCMR1 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC2M[3]	Res	Res	Res	Res	Res	Res	Res	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC4M[3]	Res	Res	Res	Res	Res	Res	Res	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC4NP	Res	CC4P	CC4E	CC3NP	Res	CC3P	CC3E	CC2NP	Res	CC2P	CC2E	CC1NP	Res	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0

Table 82. TIM2 register map and reset values (continued)

Offset	Register	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
0x24	TIMx_CNT	UIFCPY	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBL[4:0]					Res	Res	Res	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	TIMx_TISEL	Res	Res	Res	Res	TI4SEL[3:0]				Res	Res	Res	Res	Res	TI3SEL[3:0]			Res	Res	Res	Res	TI2SEL[3:0]				Res	Res	Res	Res	TI1SEL[3:0]			
	Reset value					0	0	0	0						0	0	0	0				0	0	0	0					0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

17 General-purpose timers (TIM16/17)

17.1 TIM16/17 introduction

The TIM16/17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

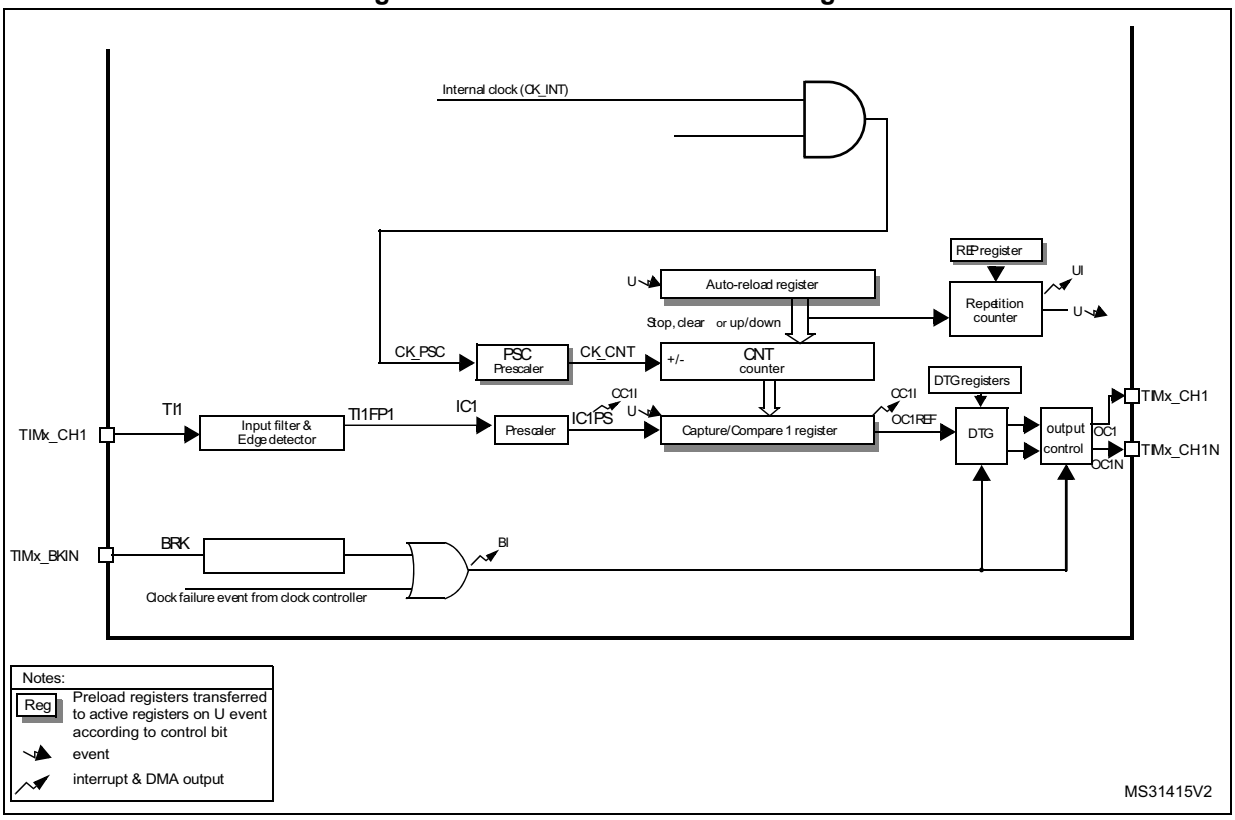
The TIM16/17 timers are completely independent, and do not share any resources.

17.2 TIM16/TIM17 main features

The TIM16 and TIM17 timers include the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - input capture
  - output compare
  - PWM generation (edge-aligned mode)
  - one-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - update: counter overflow
  - trigger event (counter start, stop, initialization or count by internal/external trigger) (only interrupt)
  - input capture
  - output compare
  - break input

Figure 70. TIM16 and TIM17 block diagram



17.3 TIM16/17 functional description

17.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description for further details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 71 and *Figure 72* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 71. Counter timing diagram with prescaler division change from 1 to 2

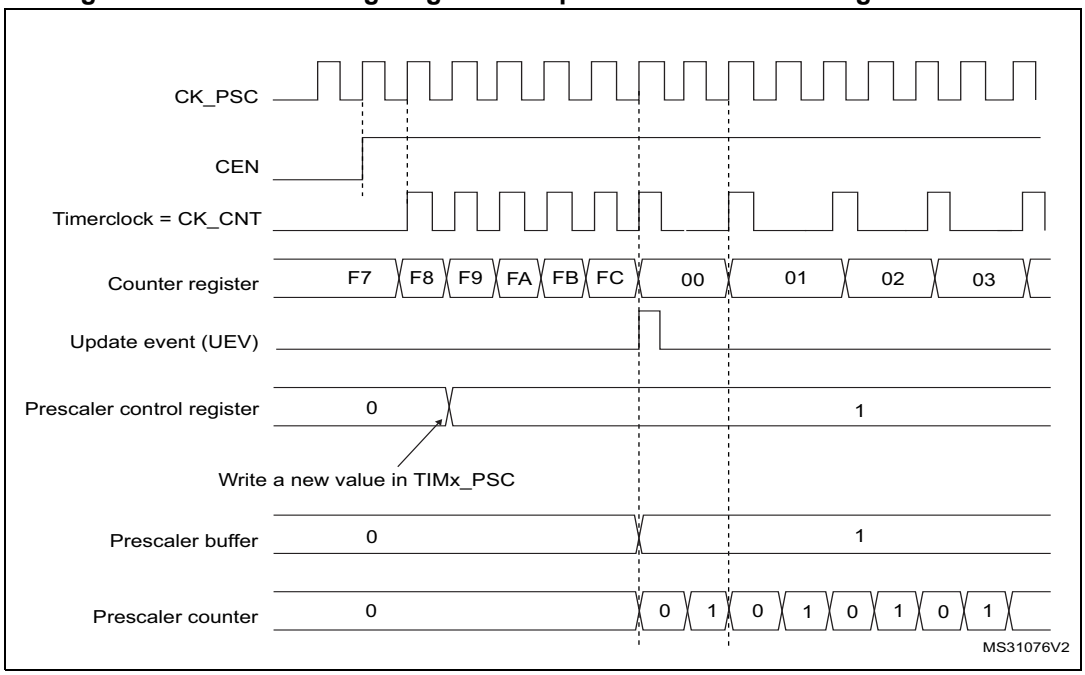
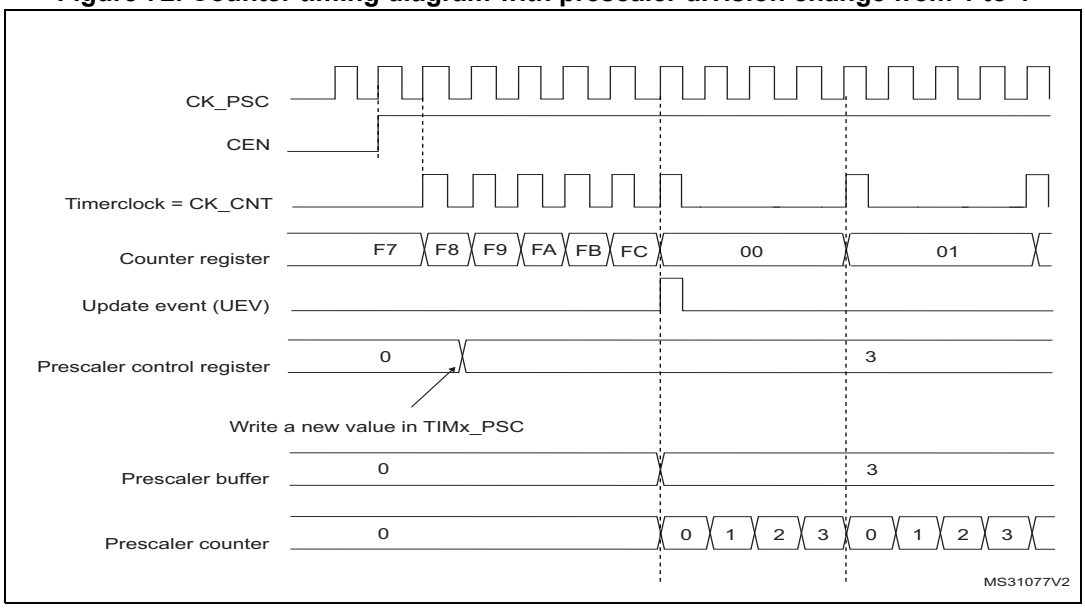


Figure 72. Counter timing diagram with prescaler division change from 1 to 4



17.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 73. Counter timing diagram, internal clock divided by 1

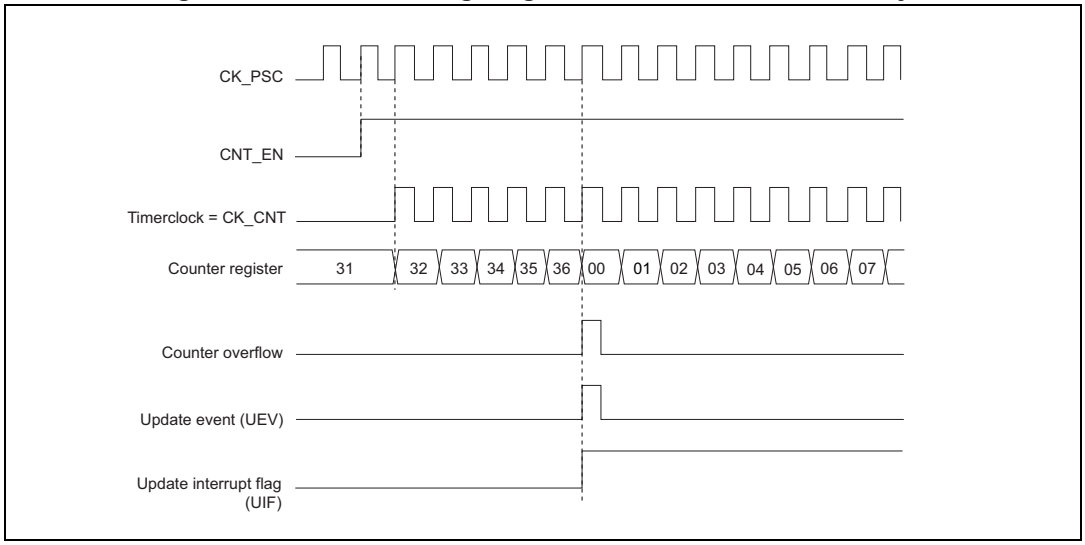


Figure 74. Counter timing diagram, internal clock divided by 2

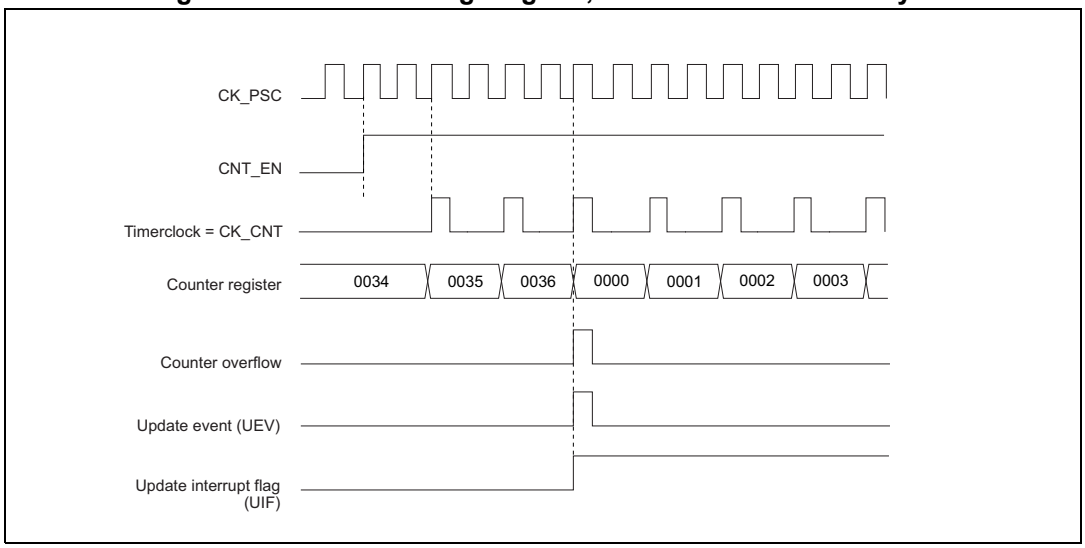


Figure 75. Counter timing diagram, internal clock divided by 4

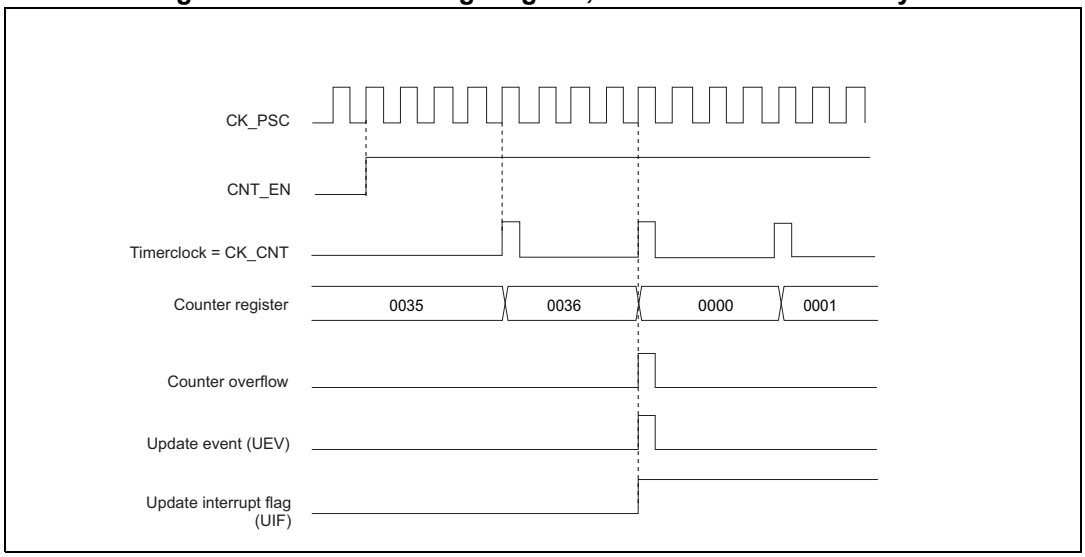


Figure 76. Counter timing diagram, internal clock divided by N

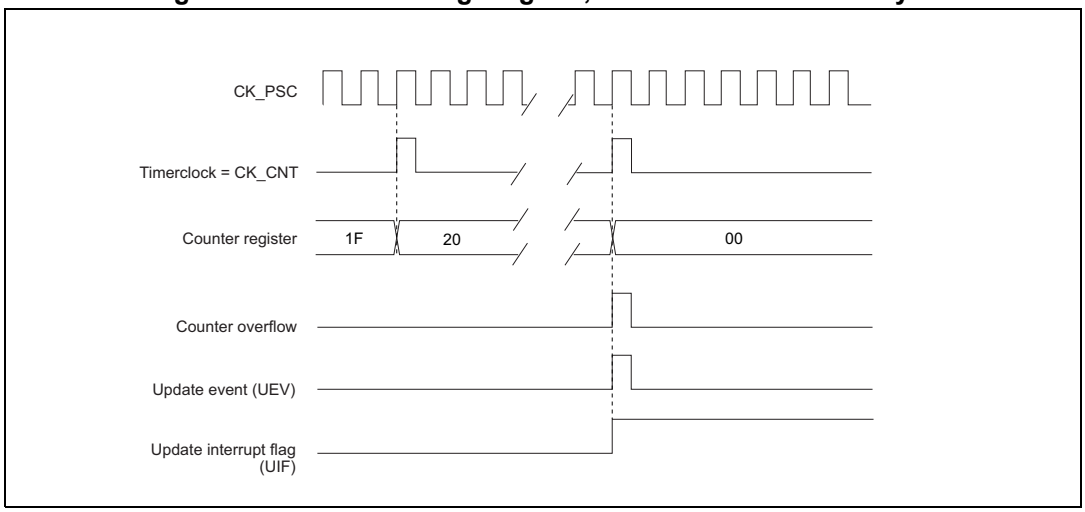


Figure 77. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

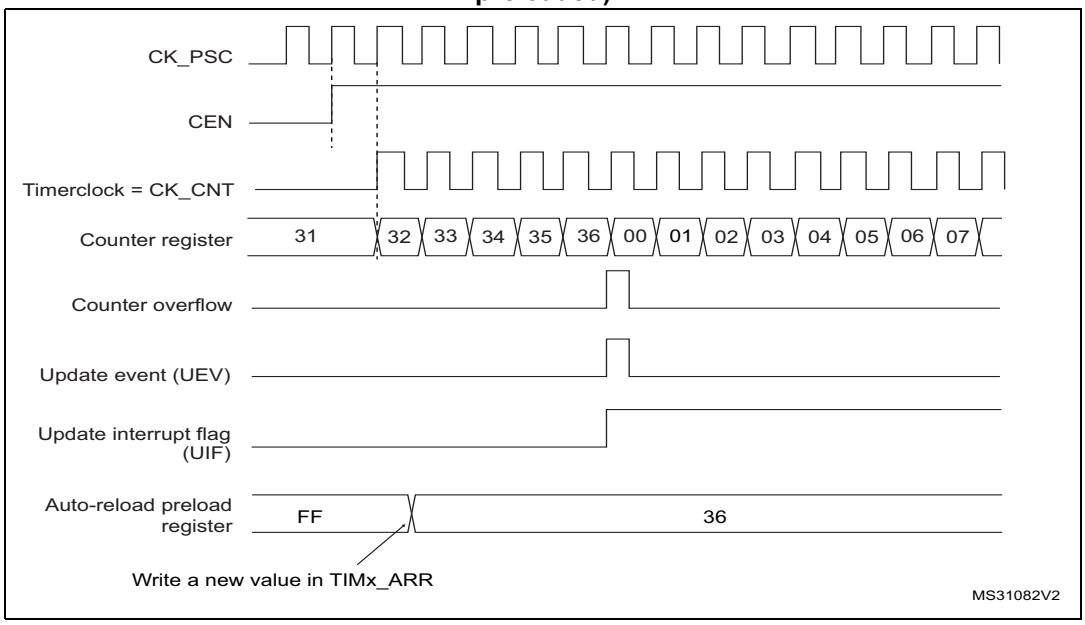
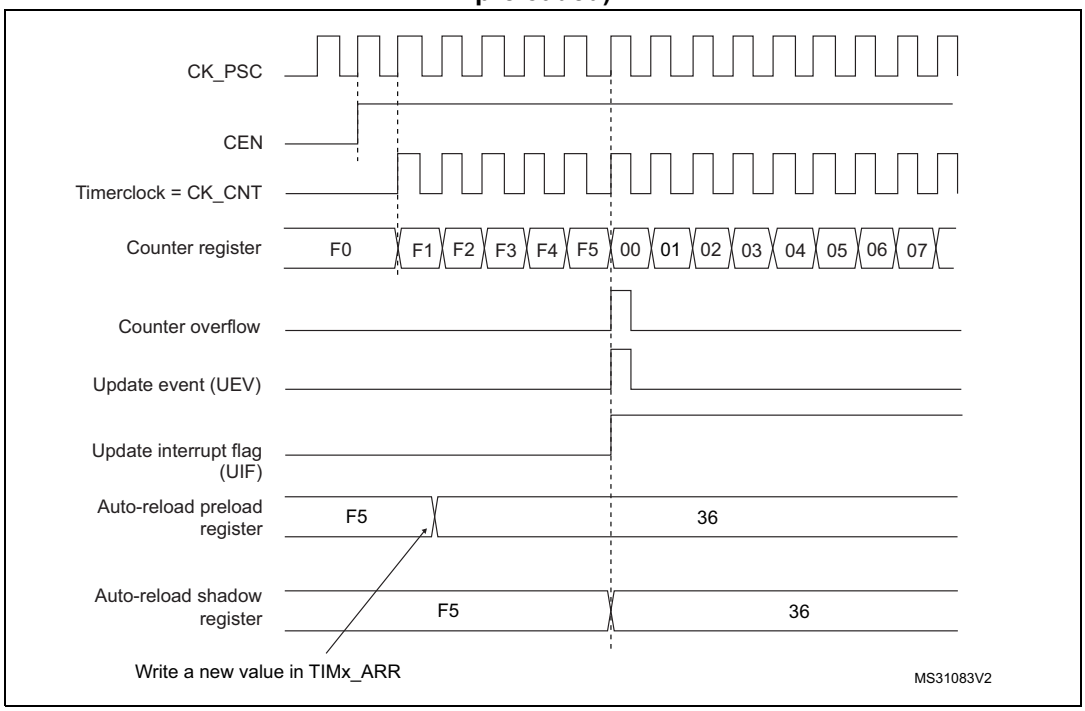


Figure 78. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



17.3.3 Repetition counter

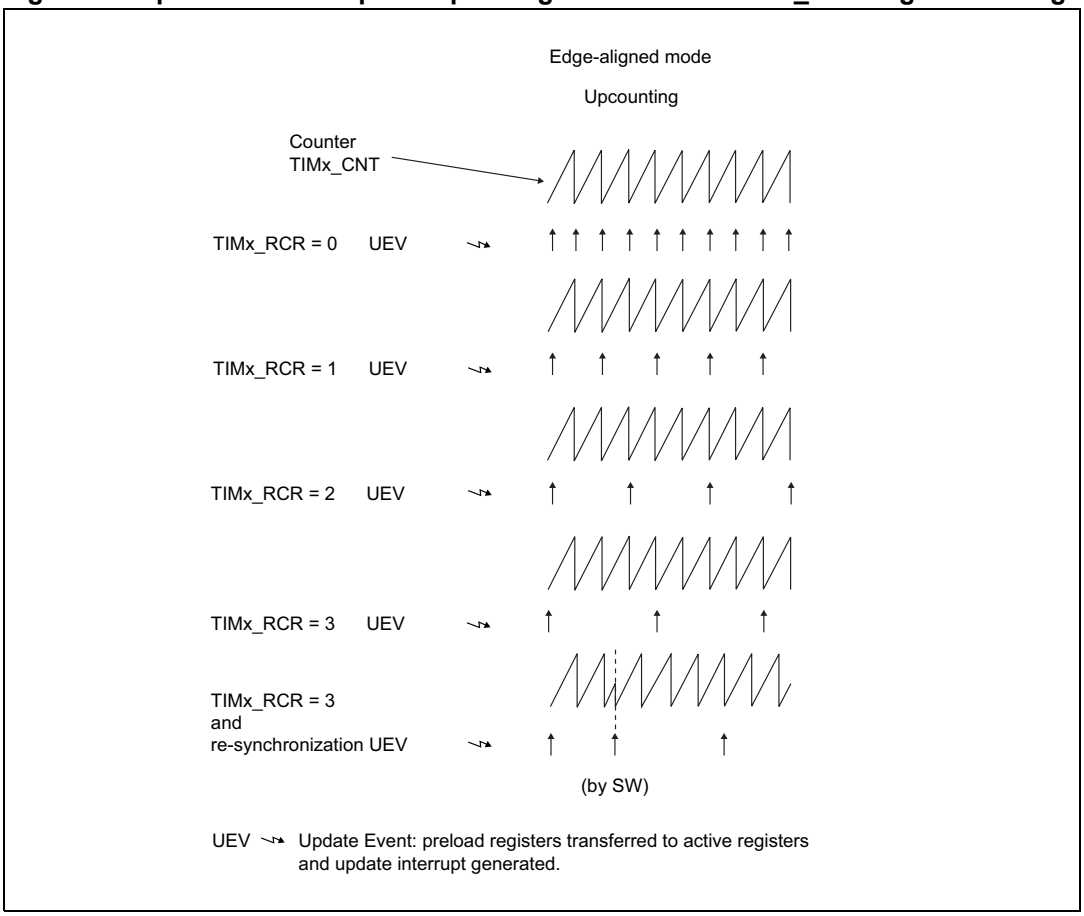
Section 17.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 79*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 79. Update rate examples depending on mode and TIMx_RCR register settings



17.3.4 Clock selection

The counter clock can be provided by the following clock sources:

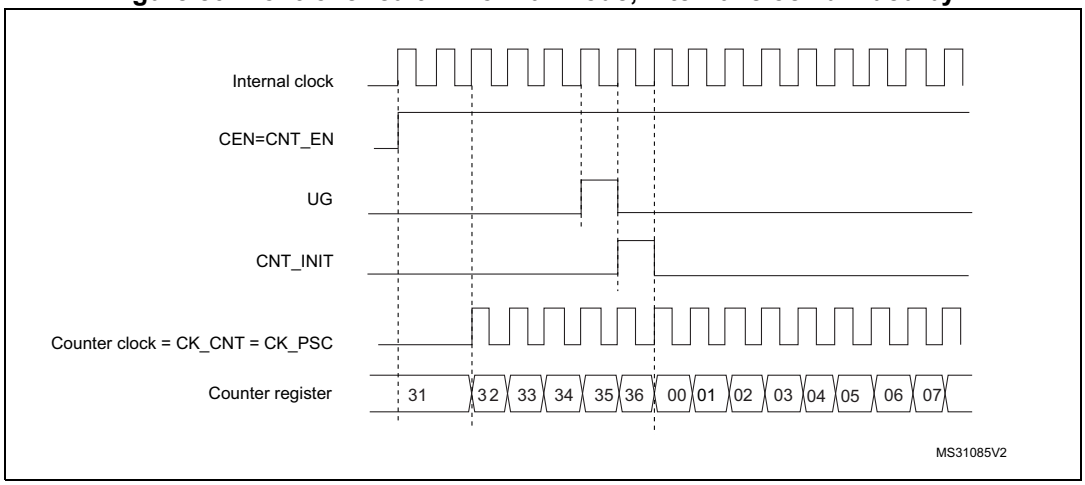
- Internal clock (CK_INT)

Internal clock source (CK_INT)

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 80 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 80. Control circuit in normal mode, internal clock divided by 1



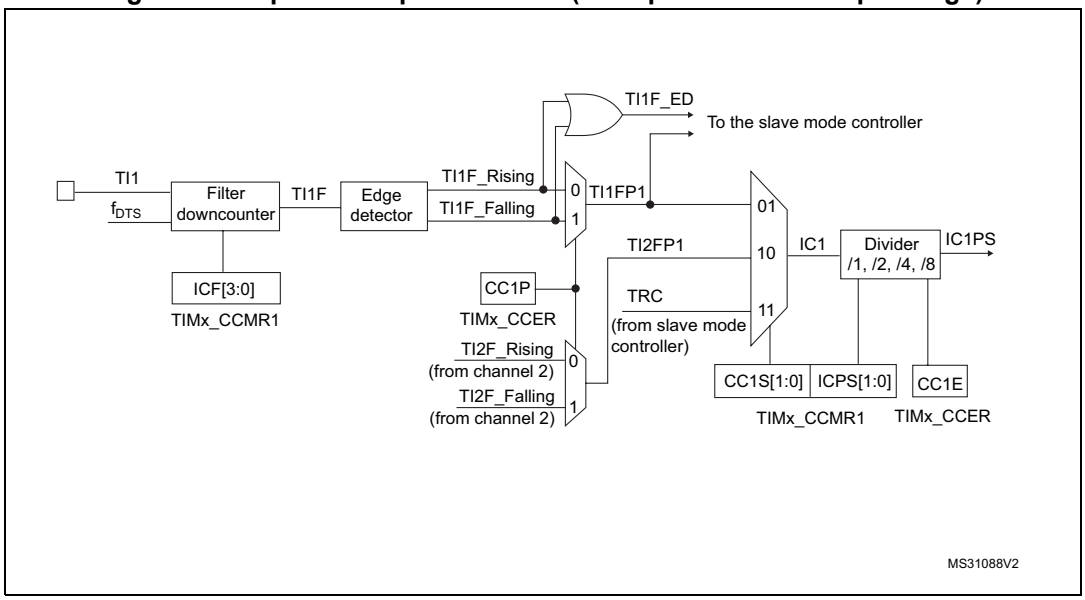
17.3.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 81 to *Figure 83* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 81. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 82. Capture/compare channel 1 main circuit

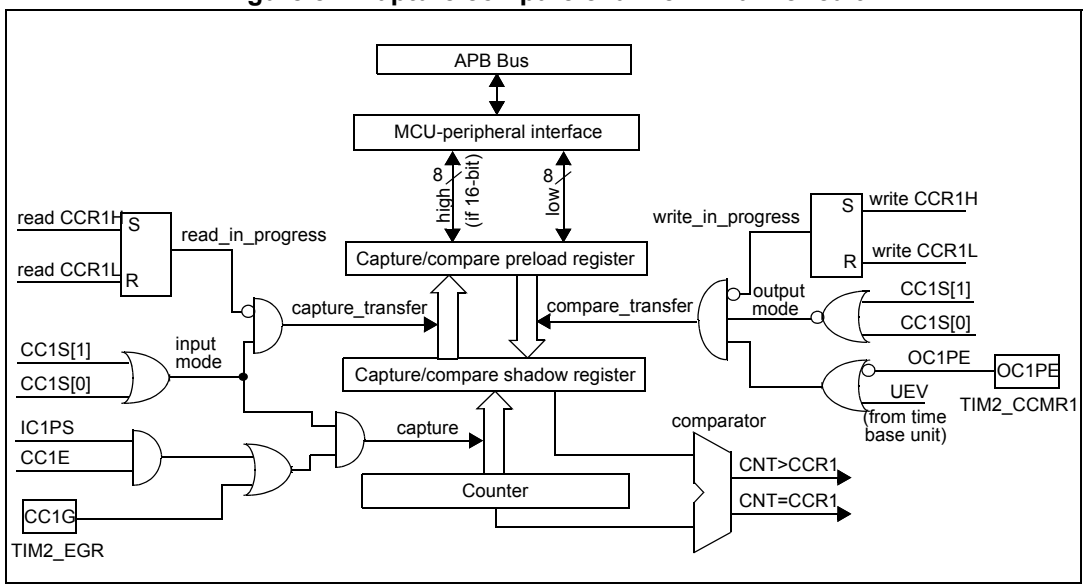
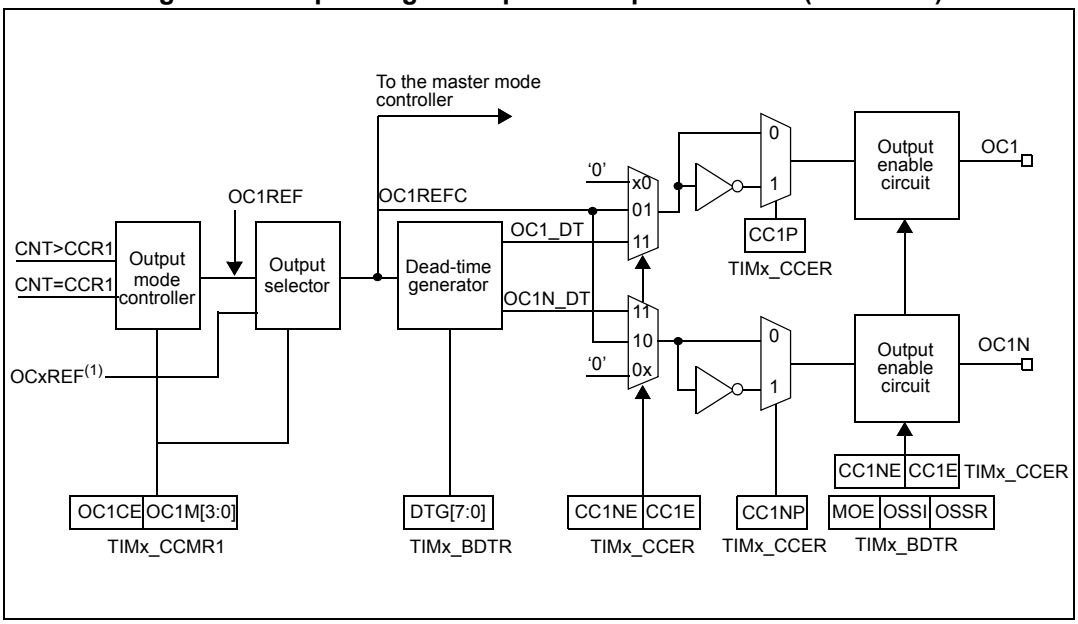


Figure 83. Output stage of capture/compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

17.3.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

3. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

17.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

17.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

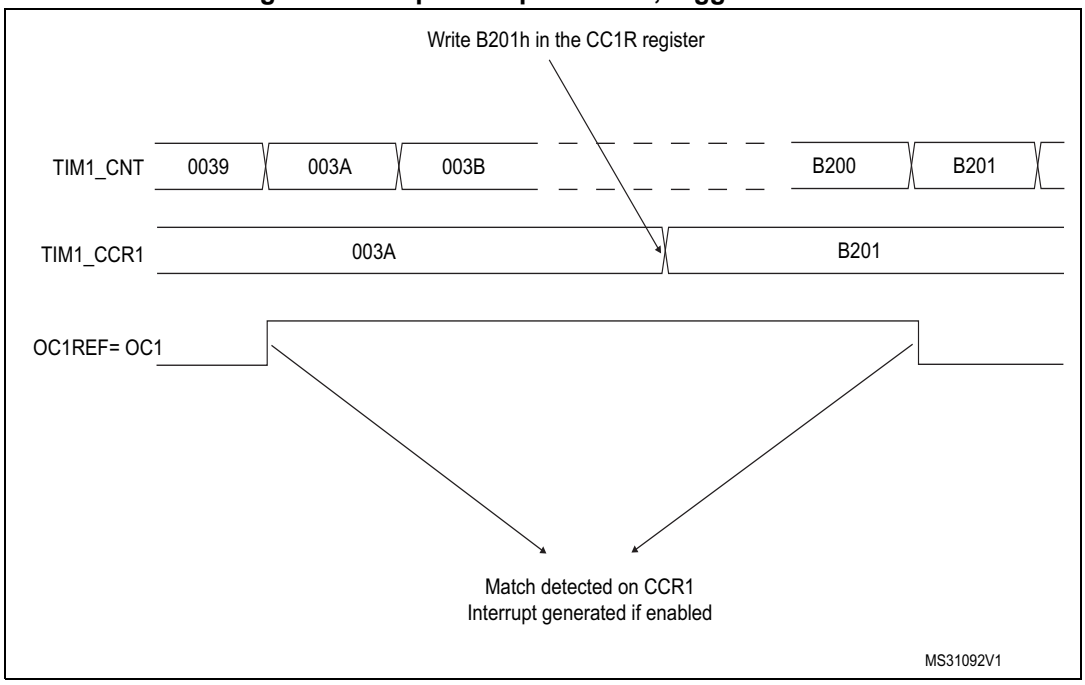
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 84*.

Figure 84. Output compare mode, toggle on OC1



17.3.9 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

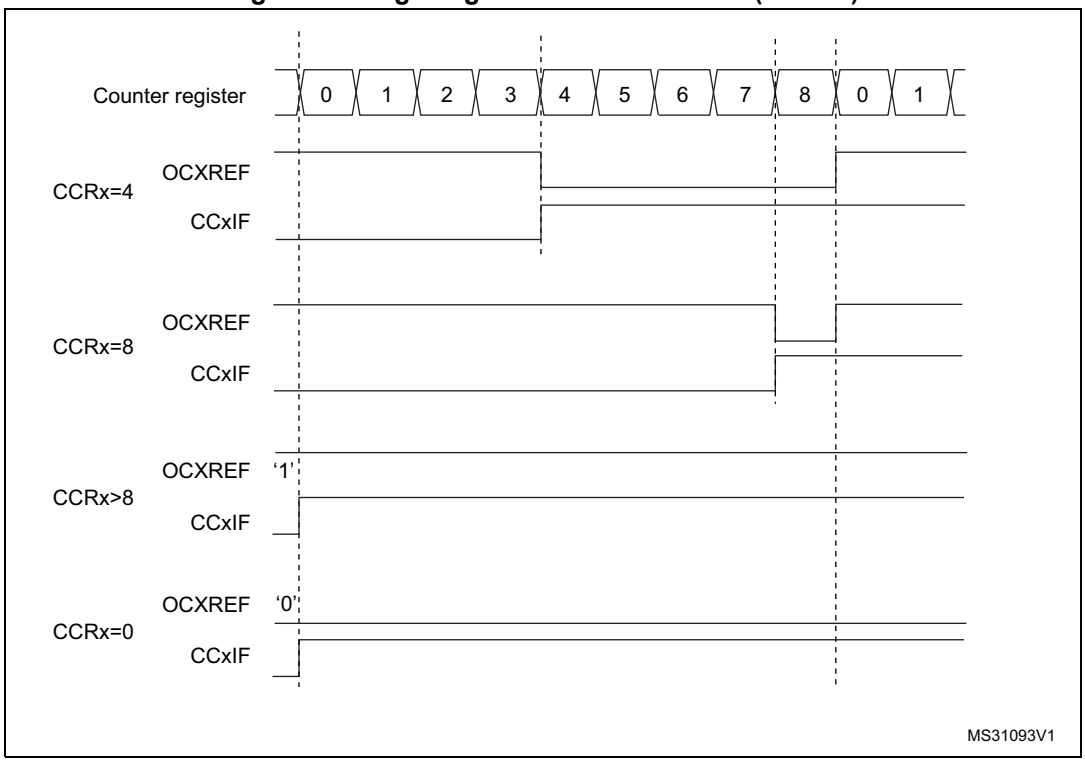
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤TIMx_CNT or TIMx_CNT ≤TIMx_CCRx (depending on the direction of the counter).

The TIM16/17 are capable of upcounting only. Refer to *Upcounting mode on page 373*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 85* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 85. Edge-aligned PWM waveforms (ARR=8)



17.3.10 Outputs and dead-time insertion

The TIM16/17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 84* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 86. Complementary output with dead-time insertion.

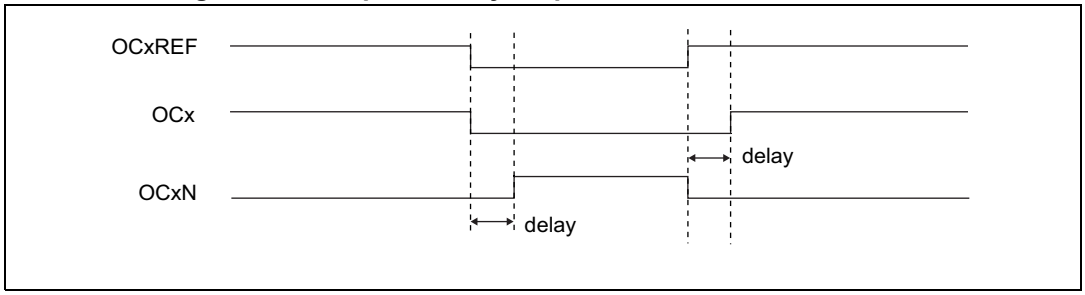


Figure 87. Dead-time waveforms with delay greater than the negative pulse.

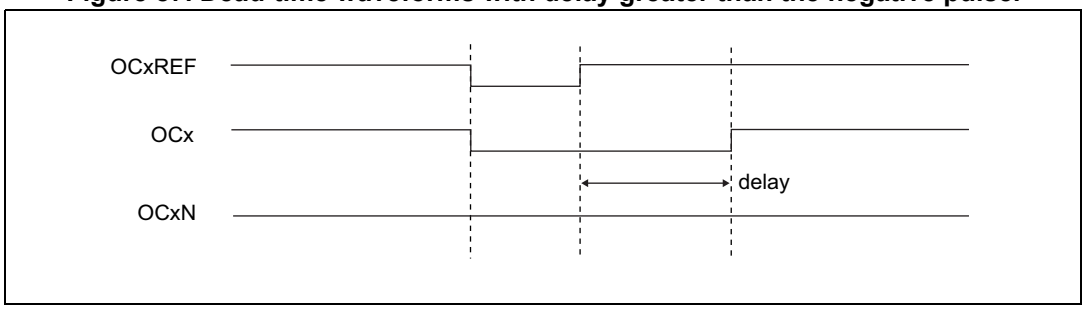
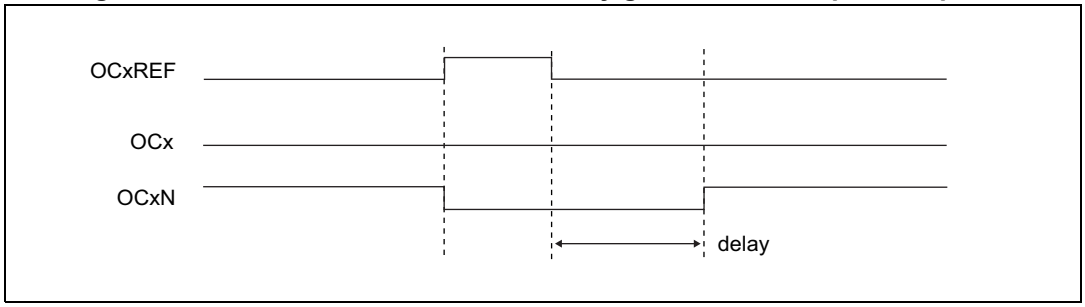


Figure 88. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 17.4.13: TIM16 and TIM17 break and dead-time register (TIMx_BDTR)* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

17.3.11 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM16/17 timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to *Table 84* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

The break is generated by the BRK inputs which has:

- Programmable polarity (BKP bit in the TIMx_BDTR register)
- Programmable enable bit (BKE bit in the TIMx_BDTR register)
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the AFIO controller (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the AFIO controller) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0 then the timer releases the enable outputs (taken over by the AFIO controller which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until you write it to '1' again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

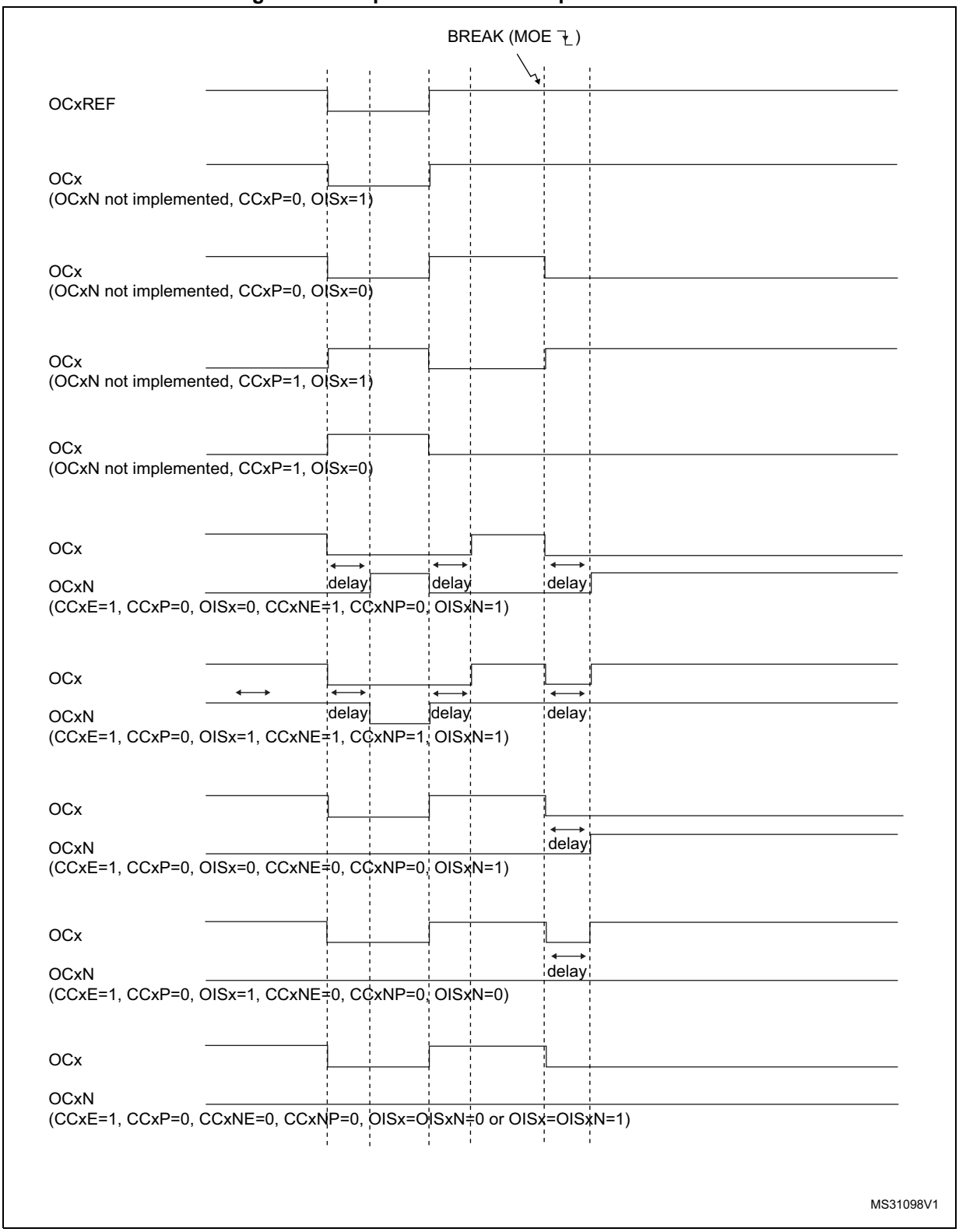
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *TIM16 and TIM17 break and dead-time register (TIMx_BDTR)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 89* shows an example of behavior of the outputs in response to a break.

Figure 89. Output behavior in response to a break



17.3.12 Bidirectional break inputs

The TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 90*.

They allow:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 83: Break protection disarming conditions*).

Table 83. Break protection disarming conditions

MOE	BKDIR	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

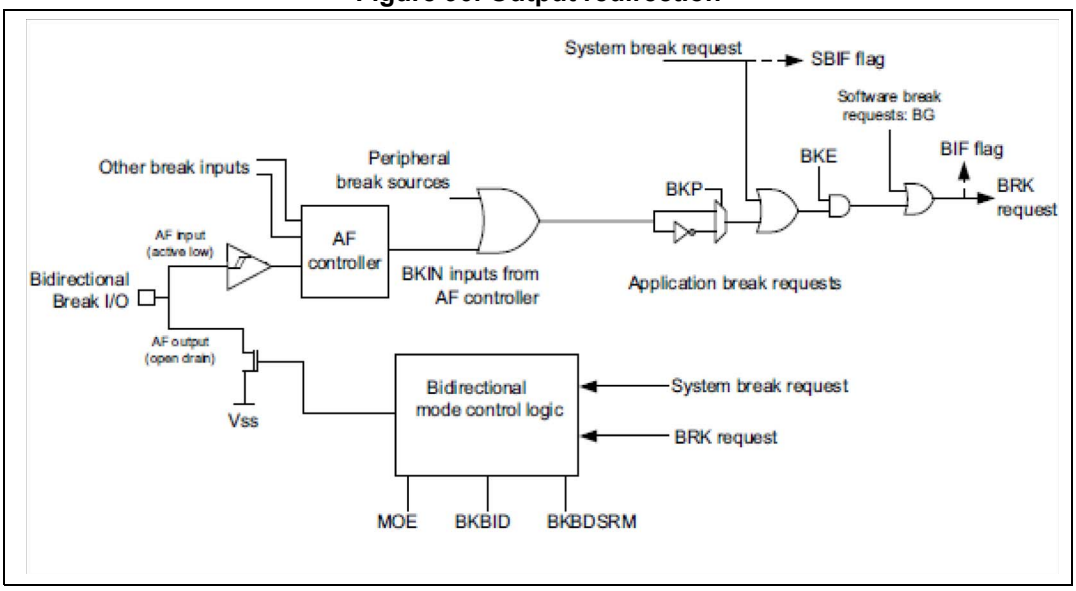
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 90. Output redirection



17.3.13 One-pulse mode

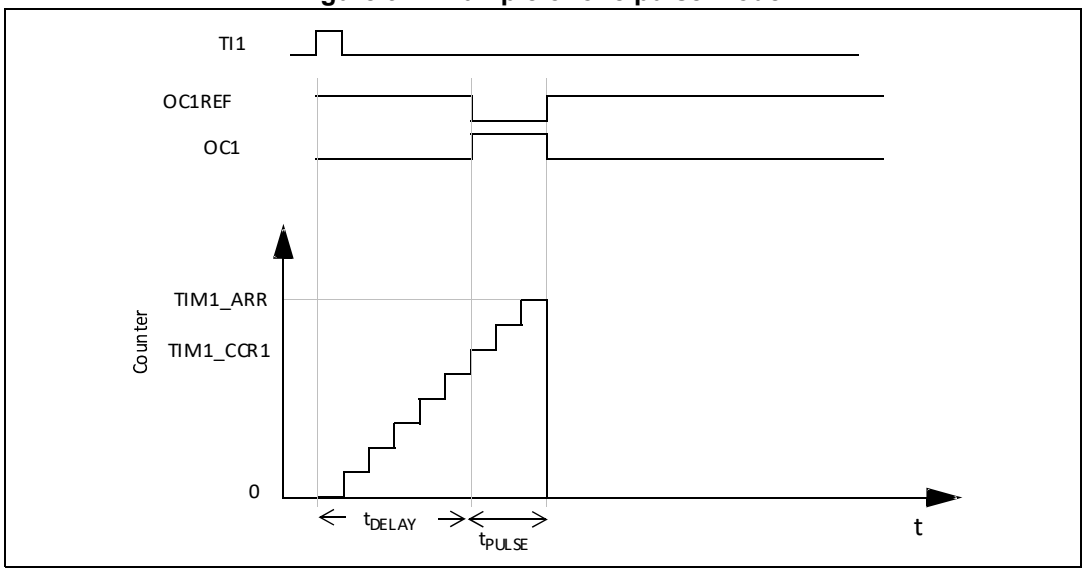
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting, the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 91. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY}.

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI1. CC1P is written to '0' in this example.

You only want 1 pulse, so you write '1' in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

17.3.14 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

17.3.15 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

17.4 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

17.4.1 TIM16 and TIM17 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	UIF REM-AP	Res	CKD[1:0]		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (TIx),
00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

17.4.2 TIM16 and TIM17 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	OIS1N	OIS1	Res	Res	Res	Res	CCDS	CCUS	Res	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

Note: This bit acts only on channels that have a complementary output.

17.4.3 TIM16 and TIM17 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CC1DE	UDE	BIE	Res	COMIE	Res	Res	Res	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bit 15:13 Reserved, must be kept at reset value.

Bit 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

17.4.4 TIM16 and TIM17 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CC1OF	Res	BIF	Res	COMIF	Res	Res	Res	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow

If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

17.4.5 TIM16 and TIM17 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	Res	COMG	Res	Res	Res	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

17.4.6 TIM16 and TIM17 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OC1M[3]
															Res.
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, always read as 0

Bit 6 **OC1M[3]:** Output Compare 1 mode (bit 3)

Bits 6:4 **OC1M[2:0]**: Output Compare 1 mode (bits 2 to 0)

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.
0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.
0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
0100: Force inactive level - OC1REF is forced low.
0101: Force active level - OC1REF is forced high.
0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.
0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.
All other values: Reserved

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

17.4.7 TIM16 and TIM17 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high

1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

*Note: **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).*

***2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high

1: OC1 active low

CC1 channel configured as input:

The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: Non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

01: Inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode.

10: Reserved, do not use this configuration.

1: Non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

*Note: **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

***2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled

1: Capture enabled

Table 84. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output Disabled (not driven by the timer: Hi-Z) OCx=CCxP, OCxN=CCxNP	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

17.4.8 TIM16 and TIM17 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

17.4.9 TIM16 and TIM17 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

17.4.10 TIM16 and TIM17 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 17.3.1: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

17.4.11 TIM16 and TIM17 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

17.4.12 TIM16 and TIM17 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1).

17.4.13 TIM16 and TIM17 break and dead-time register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	BKBID	Res	BKDSRM	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
			rw		rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, AOE, BKP, BKE, OSSI, OSSR, DTG[7:0] bits and all used bits of TIMx_AF1 register may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional
0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm
0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (opendrain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:16 Reserved, must be kept at reset value.

Bit 15 **MOE**: Main output enable
This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)
See OC/OCN enable description for more details (*Section 17.4.7: TIM16 and TIM17 capture/compare enable register (TIMx_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: **1.** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

***2.** Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

0: Break inputs (BRK) disabled

1; Break inputs (BRK) enabled

*Note: **1.** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

***2.** Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 17.4.7: TIM16 and TIM17 capture/compare enable register (TIMx_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details (*Section 17.4.7: TIM16 and TIM17 capture/compare enable register (TIMx_CCER)*).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)

1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register, BKE/BKP/AOE/BKBID/BKDSRM bits in TIMx_BDTR register and all used bits in TIMx_AF1 register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

17.4.14 TIM16 and TIM17 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBL[4:0]					Res	Res	Res	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: Reserved,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

17.4.15 TIM16 and TIM17 DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

17.4.16 TIM17 option register 1 (TIM17_OR1)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1_RMP[1:0]	
														rw	rw

Bits 1:0 **TI1_RMP[1:0]**: Timer 17 input 1 connection
This bit is set and cleared by software.
00: TIM17 TI1 is connected to GPIO
01: TIM17 TI1 is connected to RCC_LCO
1x: TIM17 TI1 is connected to RCC_MCO

17.4.17 TIM16 and TIM17 alternate function register 1(TIMx_AF1)

Address offset: 0x60

Reset value: 0x0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	BKCMP2P	BKCMP1P	BKINP	Res	Res	Res	Res	Res	Res	BKCMP2E	BKCMP1E	BKINE
				rw	rw	rw							rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity.
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active low.
1: COMP2 input is active high.
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bit 10 **BKCMP1P**: BRK COMP1 input polarity.
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active low.
1: COMP1 input is active high.
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bit 9 **BKINP**: BRK BKIN input polarity.
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active low.
1: BKIN input is active high.
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable.
This bit enables the COMP2 for the timer's BRK input. COMP2 output is ORed with the other enabled BRK sources.
0: COMP2 input disabled.
1: COMP2 input enabled.
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bit 1 **BKCMP1E**: BRK COMP1 enable.
This bit enables the COMP1 for the timer's BRK input. COMP1 output is ORed with the other enabled BRK sources.
0: COMP1 input disabled.
1: COMP1 input enabled.
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

Bit 0 **BKINE**: BRK BKIN enable.
This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is ORed with the other enabled BRK sources.
0: BKIN input disabled.
1: BKIN input enabled.
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register)

17.4.18 TIM16 input selection register (TIM16_TISEL)

Address offset: 0x68

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIMx_CH1 input
Others: Reserved

17.4.19 TIM16/TIM17 register map

TIM16 and TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 85. TIM16andTIM17 register map and reset values

Offset	Register	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
0x00	TIMx_CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIFREMA	Res	CKD [1:0]		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIMx_CR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OIS1N	OIS1	Res	Res	Res	Res	CCDS	CCUS	Res	CCPC
	Reset value																							0	0					0	0		0
0x0C	TIMx_DIER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1DE	UDE	BIE	Res	COMIE	Res	Res	Res	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	TIMx_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1OF	Res	BIF	Res	COMIF	Res	Res	Res	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	TIMx_EGR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BG	Res	COMG	Res	Res	Res	CC1G	UG
	Reset value																									0		0				0	0
0x18	TIMx_CCMR1 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OC1M[3]	Res	Res	Res	Res	Res	Res	Res	Res	Res	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	TIMx_CNT	UIFCPY or Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 85. TIM16andTIM17 register map and reset values (continued)

Offset	Register	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
0x34	TIMx_CCR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIMx_BDTR	Res	Res	Res	BKBID	Res	BKDSRM	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBL[4:0]					Res	Res	Res	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TIM17_OR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1_RMP [1:0]	
	Reset value																															0	0
0x60	TIMx_AF1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BKCMP2P	BKCMP1P	BKINP	Res	Res	Res	Res	Res	Res	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0							0	0	1
0x68	TIMx_TISEL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1SEL[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

18 Infrared interface (IRTIM)

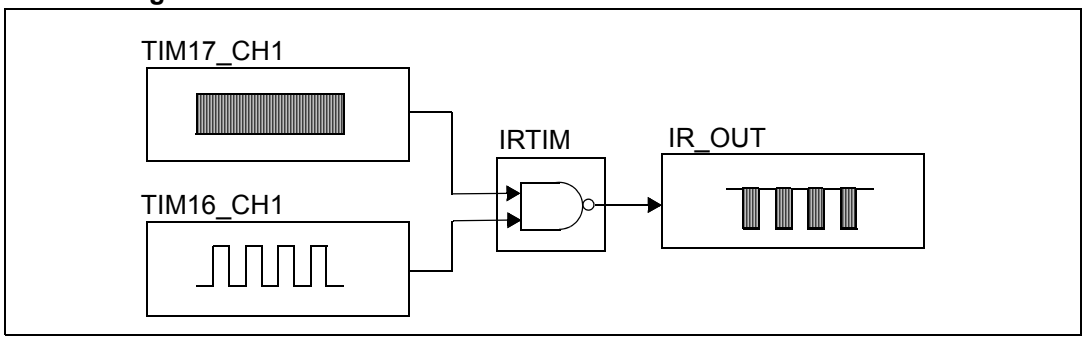
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16 and TIM17 as shown in *Figure 92*.

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16_OC1) and TIM17 channel 1 (TIM17_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

Figure 92. IR internal hardware connections with TIM16 and TIM17



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the *Section 7.4.17: GPIOA alternate function low register (GPIOA_AFRL)* or *Section 7.4.19: GPIOA alternate function high register (GPIOA_AFRH)* register by enabling the related alternate function.

The high sink LED driver capability (only available on the PA0, PA1 pin) can be activated through the I2C1_PA0_FMP or I2C1_PA1_FMP bit in the *Section 8.2.3: Fast-Mode Plus pin capability control register (I2C_FMP_CTRL)* register and used to sink the high current needed to directly control an infrared LED.

19 Real-time clock (RTC)

19.1 Introduction

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupt.

The RTC includes also a periodic programmable wakeup flag with interrupt capability.

The RTC provides an automatic wakeup to manage all low power modes.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After power-on reset, all RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low power mode or under system reset).

Note: The RTC counter does not freeze when the CPU is halted by a debugger.

19.2 RTC main features

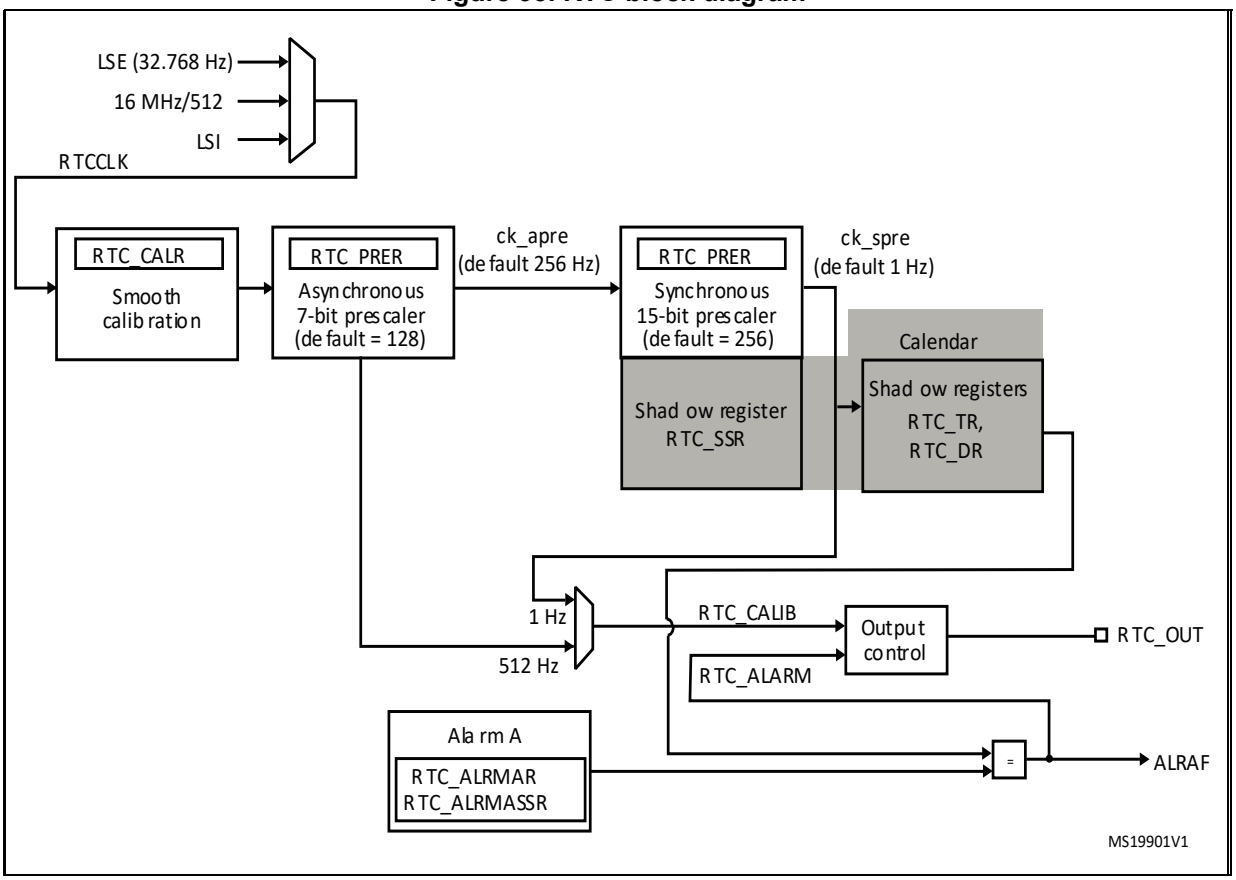
The RTC unit main features are the following (see *Figure 93: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Maskable interrupts/events:
  - Alarm A
  - Wakeup interrupt

19.3 RTC functional description

19.3.1 RTC block diagram

Figure 93. RTC block diagram



19.3.2 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and an always 32kHz equal to HSI_64M/2048 clock if HSESEL='0' else equal to HSE/1024 if HSESEL='1'. For more information on the RTC clock source configuration, refer to *Section 6: Reset and clock controller (RCC)*.

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 93: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 19.3.5: Periodic auto-wakeup* for details).

19.3.3 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every two RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see *Section 19.4*). The copy is not performed in Deepstop mode. When exiting this mode, the shadow registers are updated after up to 2 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD=0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

19.3.4 Programmable alarm

The RTC unit provides programmable alarm: Alarm A.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register. The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register. The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK0 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A (if enabled by bits OSEL[0:1] in RTC_CR register) can be routed to the RTC_ALARM output. RTC_ALARM output polarity can be configured through bit POL the RTC_CR register.

19.3.5 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE(32.768kHz), this allows to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value.When the initialization sequence is complete (see *Programming the wakeup timer on page 422*), the timer starts counting down.When the wakeup function is enabled, the down-counting remains active in low power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR2 register, it can exit the device from low power modes.

The periodic wakeup flag can be routed to the RTC_ALARM output provided it has been enabled through bits OSEL[0:1] of RTC_CR register. RTC_ALARM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low power mode (Deepstop) have no influence on the wakeup timer.

19.3.6 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After power-on reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC_ISR[13:8], and RTC_BKPxR.

1. Write ‘0xCA’ into the RTC_WPR register.
2. Write ‘0x53’ into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note: 1 After a system reset, the application can read the INITS flag in the RTC_ISR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its power-on reset default value (0x00).

2 To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ISR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms.

1. Clear ALRAE in RTC_CR to disable Alarm A.
2. Program the Alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable Alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ISR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR).Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting.

19.3.7 Reading the calendar

- When BYPSHAD control bit is cleared in the RTC_CR register:

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be equal to or greater than seven times the f_{RTCCLK} RTC clock frequency. This ensures a secure behavior of the synchronization mechanism.

If the APB0 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB0 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ISR register each time the calendar registers are copied into the RTC_TR and RTC_DR shadow registers. The copy is performed every two RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 2 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low power mode (Deepstop), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 421*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 19.3.9: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

- When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers):

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low power modes (Deepstop), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD=1, instructions which read the calendar registers require one extra APB cycle to complete.

19.3.8 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and the RTC status register (RTC_ISR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a power-on reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR) , the RTC backup registers (RTC_BKPxR), the wakeup timer register (RTC_WUTR), the Alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).

In addition, the RTC keeps on running under system reset if the reset source is different from the power-on reset one. When a power-on reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

19.3.9 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR .

RTC_SSR contains the value of the synchronous prescaler’s counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler’s output at 1 Hz. In this way, the frequency of the

asynchronous prescaler's output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (PREDIV_S + 1) seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow occurs.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

19.3.10 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjusments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, calib_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting SMC[2] to 1 causes four additional cycles to be masked
- and so on up to SMC[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1]=1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); SMC[2]=1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to SMC[8]=1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm ($511/(2^{20}+511)$) with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm ($512/(2^{20}-512)$). Setting CALP to '1' effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [\ 1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)\ ]$$

Calibration when PREDIV_A<3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the

calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [\ 1 + (256 - CALM) / (2^{20} + CALM - 256)\ ]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Note: The case PREDIV_A=2 (asynchronous prescaler divides by 3) seems unlikely to be useful unless the nominal input frequency is a multiple of 3. For example, if RTCCLK is nominally 98304Hz (32768Hz x 3), setting PREDIV_S to 32759 rather than 32767 (8 less) would render the above formula valid.

Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8- second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stucked at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ISR/INITF=0, by using the follow process:

1. Poll the RTC_ISR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR,if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

Note: RECALPF then becomes '0' automatically. Note that RECALPF can stay at '1' for as long as 4 ck_apre cycles plus 2 system clock cycles after writing to RTC_CALR. During initialization mode (RTC_ISR/INIT=1), RECALPF can stay at '1' indefinitely.

19.3.11 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output.

Note: This RTC_CALIB information is output on the RTC_OUT I/O signal if the I/O is programmed with the associated AFx mode (see Section 4: I/O operating modes).

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The RTC_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and "PREDIV_S+1" is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the RTC_CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV\_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

19.3.12 Alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm alternate function output RTC_ALARM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The polarity of the output is determined by the POL control bit in RTC_CR so that the opposite of the selected flag bit is output when POL is set to 1.

Note: 1 The RTC_ALARM is output on the RTC_OUT I/O signal if the I/O is programmed with the associated AFx mode (see Section 4 on page 53)

2 Once the RTC_ALARM output is enabled, it has priority over RTC_CALIB (COE bit is don't care and must be kept cleared), which means the RTC_OUT I/O outputs the RTC_ALARM.

19.4 RTC low power modes

The RTC is able to run in Deepstop mode and generate a wakeup event to wake the device through RTC alarm and RTC wakeup root cause.

Note: The software must clear the RTC_ISR.WUTF flag in the RTC after a wakeup. Not doing this prevents re-entry into low power mode. The PWRC block only mirrors the RTC wakeup signal in its own wakeup flag register.

19.5 RTC interrupts

All RTC interrupts are combined and connected to the NVIC controller. Refer to *Table 6: Interrupt vectors.*

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the RTC_ALARM IRQ channel in the NVIC.
2. Configure the RTC to generate RTC alarms (Alarm A).

To enable the Wakeup timer interrupt, the following sequence is required:

1. Configure and Enable the RTC IRQ channel in the NVIC.
2. Configure the RTC to detect the WUT event.

19.6 RTC registers

Refer to *Section 1.5* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

19.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 421* and *Reading the calendar on page 422*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 421*.

Address offset: 0x00

Power-on reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0 . Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31-23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bit 16:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bit 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bit 3:0 **SU[3:0]**: Second units in BCD format

19.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 421* and *Reading the calendar on page 422*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 421*.

Address offset: 0x04

Power-on reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0 . Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

19.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WUTIE	Res	ALRAIE	Res.	WUTE	Res	ALRAE	Res.	FMT	BYPS HAD	Res.	Res.	WUCKSEL[2:0]		
	rw		rw		rw		rw		rw	rw			rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 23 **COE**: Calibration output enable

This bit enables the RTC_CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to RTC_ALARM output
00: Output disabled
01: Alarm A output enabled
10: Reserved
11: Wakeup output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of RTC_ALARM output
0: The pin is high when ALRAF/WUTF is asserted (depending on OSEL[1:0])
1: The pin is low when ALRAF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL** : Calibration output selection

When COE=1, this bit selects which signal is output on RTC_CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A=127 and PREDIV_S=255). Refer to *Section 19.3.11: Calibration clock output*

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bitis always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 Reserved, must be kept at reset value

Bit 14 **WUTIE**: Wakeup timer interrupt enable
0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 Reserved, must be kept at reset value

Bit 10 **WUTE**: Wakeup timer enable
0: Wakeup timer disabled
1: Wakeup timer enabled

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ALRAE:** Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **WUCKSEL[2:0]**: Wakeup clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected
11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value (see note below)

Note: 1 Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).

2 WUT = Wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

3 Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1.

4 It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

5 ADD1H and SUB1H changes are effective in the next second.

6 This register is write protected. The write access procedure is described in RTC register write protection on page 421.

19.6.4 RTC initialization and status register (RTC_ISR)

This register is write protected (except for RTC_ISR[-17:8] bits). The write access procedure is described in *RTC register write protection on page 421*.

Address offset: 0x0C

Reset value: 0x0000 0007

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			Res.	Res.	WUTF		ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF		ALRAWF
					rc_w0		rc_w0	rw	r	rc_w0	r	rc_w0	r		r

Bits 31:18 Reserved, must be kept at reset value

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to ‘1’ when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to ‘0’. Refer to *Re-calibration on-the-fly*.

Bit 15 Reserved, must be kept at reset value

Bit 14 Reserved, must be kept at reset value

Bit 13 Reserved, must be kept at reset value

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.
This flag is cleared by software by writing 0.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 8 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm A register (RTC_ALRMAR).
This flag is cleared by software by writing 0.

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed.

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow regsiter mode (BYPSHAD=1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (power-on reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**:Shift operation pending

0: No shift operation is pending

1: A shift operation is pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when the wakeup timer values can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wakeup timer configuration update not allowed

1: Wakeup timer configuration update allowed.

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

Note: 1 The bits ALRAF, WUTF are cleared 2 APB clock cycles after programming them to 0.

19.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 421*

This register is write protected. The write access procedure is described in *RTC register write protection on page 421*.

Address offset: 0x10

Power-on reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

19.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ISR.

This register is write protected. The write access procedure is described in *RTC register write protection on page 421*.

Address offset: 0x14

Power-on reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs (WUT+1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] =011 (RTCCLK/2) is forbidden.

19.6.7 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 421*.

Address offset: 0x1C

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don't care in Alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format.

Bits 27:24 **DU[3:0]**: Date units or day in BCD format.

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don't care in Alarm A comparison

Bit 22 **PM:** AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don't care in Alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in Alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

19.6.8 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

19.6.9 RTC sub second register (RTC_SSR)

Address offset: 0x28

Power-on reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0 . Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits31:16 Reserved, must be kept at reset value

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler's counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

19.6.10 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 421*.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 31:15 Reserved, must be kept at reset value

Bits 14:0 **SUBFS**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

The value which is written to SUBFS is added to the synchronous prescaler's counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by :

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))) .

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.

Refer to *Section 19.3.9: RTC synchronization*.

19.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 421*.

Address offset: 0x3C

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:16 Reserved, must be kept at reset value

Bit 15 CALP: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency incresed by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 * CALP) - CALM.

Refer to *Section 19.3.10: RTC smooth digital calibration*.

Note:

Bit 14 **CALW8:** Use an 8-second calibration cycle period

When CALW8 is set to '1' , the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stucked at "00" when CALW8='1'. Refer to Section 19.3.10: RTC smooth digital calibration.

Bit 13 **CALW16:** Use a 16-second calibration cycle period

When CALW16 is set to '1' , the 16-second calibration cycle period is selected.This bit must not be set to '1' if CALW8=1.

Note: CALM[0] is stucked at '0' when CALW16='1'. Refer to Section 19.3.10: RTC smooth digital calibration.

Bits 12:9 Reserved, must be kept at reset value

Bits 8:0 CALM[8:0]: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 19.3.10: RTC smooth digital calibration*.

19.6.12 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 421*

Address offset: 0x44

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31:28 Reserved, must be kept at reset value.

Bit 27:24 MASKSS[3:0]: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[14:1] are don't care in Alarm A comparison. Only SS[0] is compared.
2: SS[14:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.
3: SS[14:3] are don't care in Alarm A comparison. Only SS[2:0] are compared.
...
12: SS[14:12] are don't care in Alarm A comparison. SS[11:0] are compared.
13: SS[14:13] are don't care in Alarm A comparison. SS[12:0] are compared.
14: SS[14] is don't care in Alarm A comparison. SS[13:0] are compared.
15: All 15 SS bits are compared and must match to activate alarm.
The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bit 23:15 Reserved, must be kept at reset value.

Bit 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

19.6.13 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x54

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 BKP[31:0]

The application can write or read data to and from these registers.
They are powered-on by VDD12o so they are retained during Deepstop mode.
The application can write or read data to and from these registers. This register is reset on PORESETn only.

19.6.14 RTC register map

Table 86. RTC register map and reset values

Offset	Register	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
0x00	**RTC_TR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	**RTC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	Res.	WUTIE	Res.	ALRAIE	Res.	WUTE	Res.	ALRAE	Res.	FMT	BYPSHAD	Res.	Res.	WUCKSEL[2:0]		
	Reset value									0	0	0	0		0	0	0		0		0		0		0		0	0	0		0	0	0
0x0C	**RTC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF						WUTF		ALRAF	INIT	INITF	RSF	INITS	SHPF	WUT WF		ALRAWF
	Reset value																0						0		0	0	0	0	0	0	1		1
0x10	**RTC_PRER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							Res.	PREDIV_S[14:0]														
	Reset value										1	1	1	1	1	1	1		0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	**RTC_ALRMAR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	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
0x24	**RTC_WPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_SSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**RTC_TSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	**RTC_TSDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	**RTC_TSSSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0

Table 86. RTC register map and reset values (continued)

Offset	Register	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
0x40	**RTC_TAMPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value								0	0	0	0	0	0	0	0	0																
0x40	Reserved																																
0x44	RTC_ ALRMASSR	Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50 to 0x54	**RTC_BKP0R**	BKP[31:0]																															
	Reset value	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
	to **RTC_BKP1R**	BKP[31:0]																															
	Reset value	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

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

20 Independent watchdog (IWDG)

20.1 Introduction

The devices feature an embedded watchdog peripheral which offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral serves to detect and resolve malfunctions due to software failure, and to trigger system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by slow speed clock source (see *Section 6.6.2: Clock configuration register (RCC_CFGR)*) and thus stays active even if the main clock fails.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints.

20.2 IWDG main features

- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Deepstop mode)
- Conditional Reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 000h
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

20.3 IWDG functional description

Figure 94 shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0x0000 CCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

20.3.1 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the IWDG_WINR register.

If the reload operation is performed while the counter is greater than the value stored in the window register (IWDG_WINR), then a reset is provided.

The default value of the IWDG_WINR is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the IWDG_RLR value and ease the cycle number calculation to generate the next reload.

Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG_KR register.
2. Enable register access by writing 0x0000 5555 in the IWDG_KR register.
3. Write the IWDG prescaler by programming IWDG_PR from 0 to 7.
4. Write the reload register (IWDG_RLR).
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the window register IWDG_WINR. This automatically refreshes the counter value IWDG_RLR.

Note: Writing the window value allows to refresh the Counter value by the RLR when IWDG_SR to set to 0x0000 0000.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable register access by writing 0x0000 5555 in the IWDG_KR register.
2. Write the IWDG prescaler by programming IWDG_PR from 0 to 7.
3. Write the reload register (IWDG_RLR).
4. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
5. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).
6. Enable the IWDG by writing 0x0000 CCCC in the IWDG_KR.

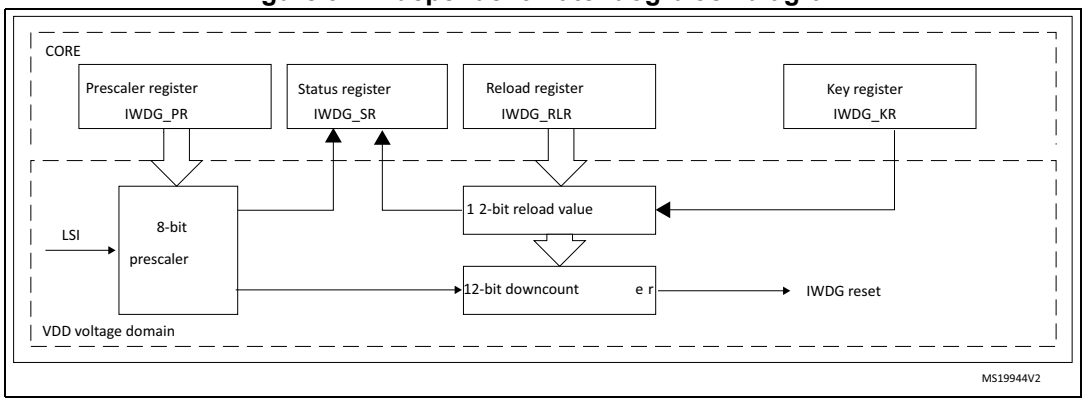
20.3.2 Register access protection

Write access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers is protected. To modify them, you must first write the code 0x0000 5555 in the IWDG_KR register. A write access to this register with a different value breaks the sequence and register access is protected again. This implies that it is the case of the reload operation (writing 0x0000 AAAA).
A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is on going.

20.3.3 Debug mode

No specific debug mode implemented in STM32WB09xE. The timer goes on counting even when the CPU is halted by the debugger.

Figure 94. Independent watchdog block diagram



Note: The watchdog is implemented in the VDD12o power domain that is still functional in Deepstop mode.

20.4 IWDG registers

Refer to *Section 1.5: Acronyms* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

20.4.1 Key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enables access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see *Section 20.3.2*)

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

20.4.2 Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see *Section 20.3.2*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG_SR register is reset.

20.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Section 20.3.2*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet *[1]* for the timeout information.

The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

20.4.4 Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

20.4.5 Window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected see *Section 20.3.2*. These bits contain the high limit of the window value to be compared to the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG_SR register is reset.

Note: If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.

20.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

Table 87. IWDG register map and reset values

Offset	Register	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
0x00	**IWDG_KR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
	Reset value																														0	0	0
0x08	**IWDG_RLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
	Reset value																														0	0	0
0x10	**IWDG_WINR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

21 Inter-integrated circuit (I2C) interface

21.1 Introduction

The I^2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^2C bus. It provides multicontroller capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

21.2 I2C main features

- I^2C bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available depending on the product implementation (see *Section 21.3: I2C implementation*):

- SMBus specification rev 2.0 compatibility:
  - Hardware PEC (Packet Error Checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and Device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.1 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming

Note: For the Fast-Mode Plus mode, it is strongly recommended to use I2C pins mentioned as open-drain capable (embedding a 10ns/50ns filter).

21.3 I2C implementation

This manual describes the full set of features implemented in I2C1.

Table 88. I2C implementation

I2C features(1)	I2C1
7-bit addressing mode	X
10-bit addressing mode	X
Standard-mode (up to 100 kbit/s)	X
Fast-mode (up to 400 kbit/s)	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X
Independent clock	X
SMBus	X

1. X = supported.

21.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I^2C bus.

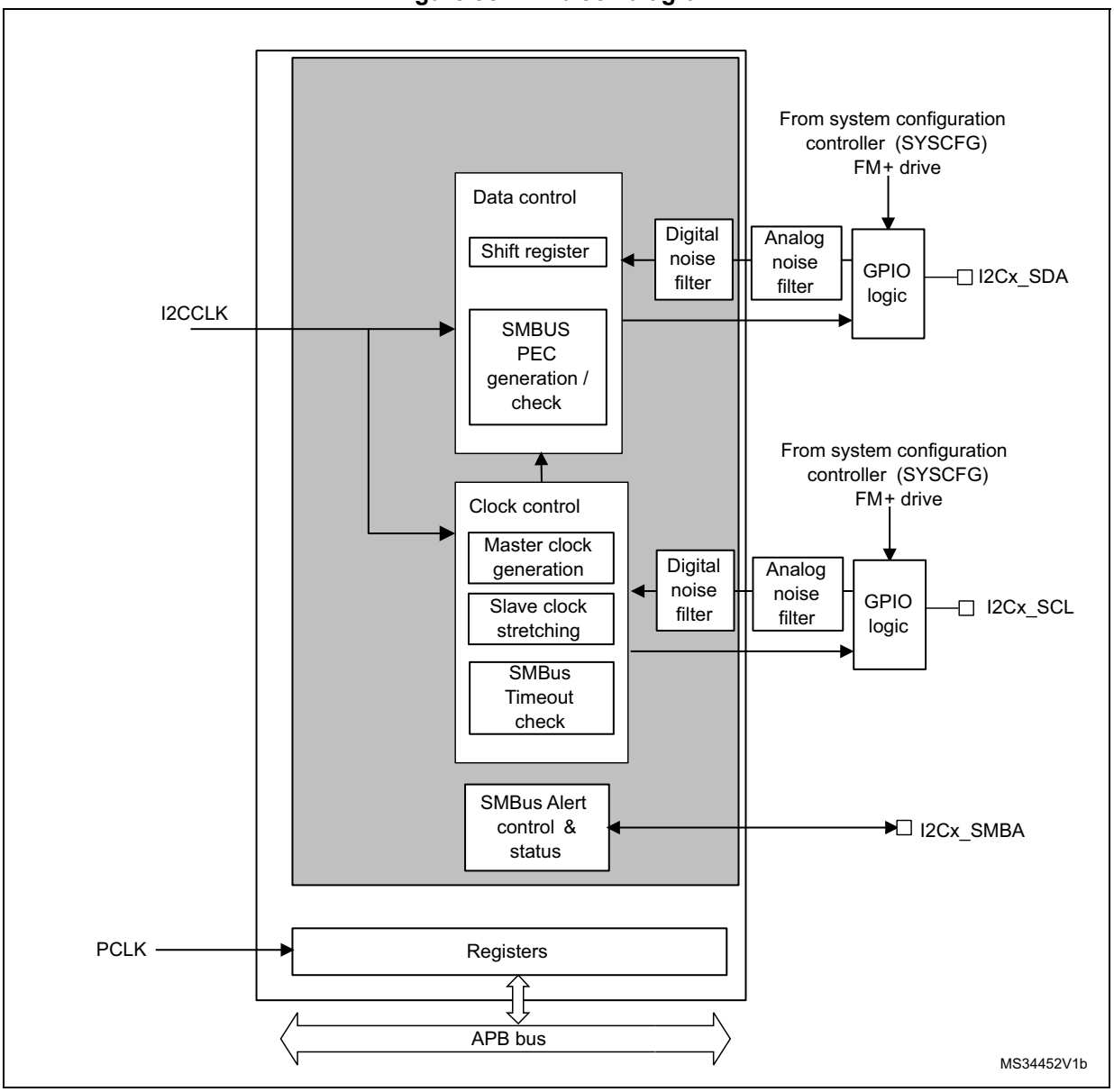
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

21.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 95*.

Figure 95. I2C block diagram



The I2C is clocked by an independent clock source which allows to the I2C to operate independently from the PCLK frequency.

This independent clock source is a fixed 16 MHz clock. Refer to *Section 6: Reset and clock controller (RCC)* for more details.

I2C I/Os support 20 mA output current drive for Fast-mode Plus operation. This is enabled by setting the driving capability control bits for SCL and SDA in *Section 8.2.3: Fast-Mode Plus pin capability control register (I2C_FMP_CTRL)*.

21.4.2 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (tLOW - tfilters) / 4$ and $t_{I2CCLK} < tHIGH$

with:

t_{LOW}: SCL low time and tHIGH : SCL high time

$t_{filters}$: when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is DNF x t_{I2CCLK}.

The PCLK clock period t_{PCLK} must respect the following condition:

$t_{PCLK} < 4/3\ tSCL$

with t_{SCL}: SCL period

21.4.3 Mode selection

The interface can operate in one of the four following modes:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, it operates in target mode. The interface automatically switches from target to controller when it generates a START condition, and from controller to target if an arbitration loss or a STOP generation occurs, allowing multicontroller capability.

Communication flow

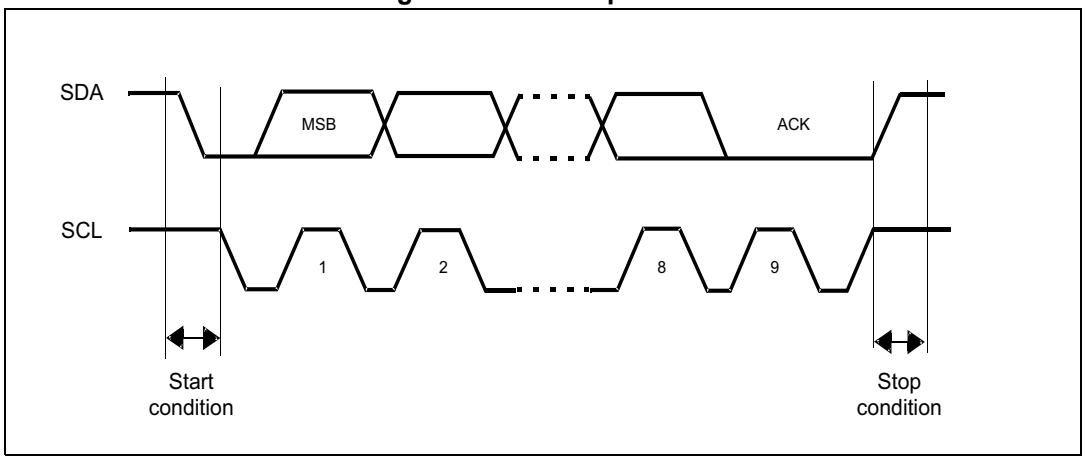
In Controller mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In Target mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Controller mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 96. I²C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

21.4.4 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller (refer to *Section 6: Reset and clock controller (RCC)*).

Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE=0), the I^2C performs a software reset. Refer to *Section 21.4.5: Software reset* for more details.

Noise filters

Before you enable the I2C peripheral by setting the PE bit in I2C_CR1 register, you must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I^2C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. You can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

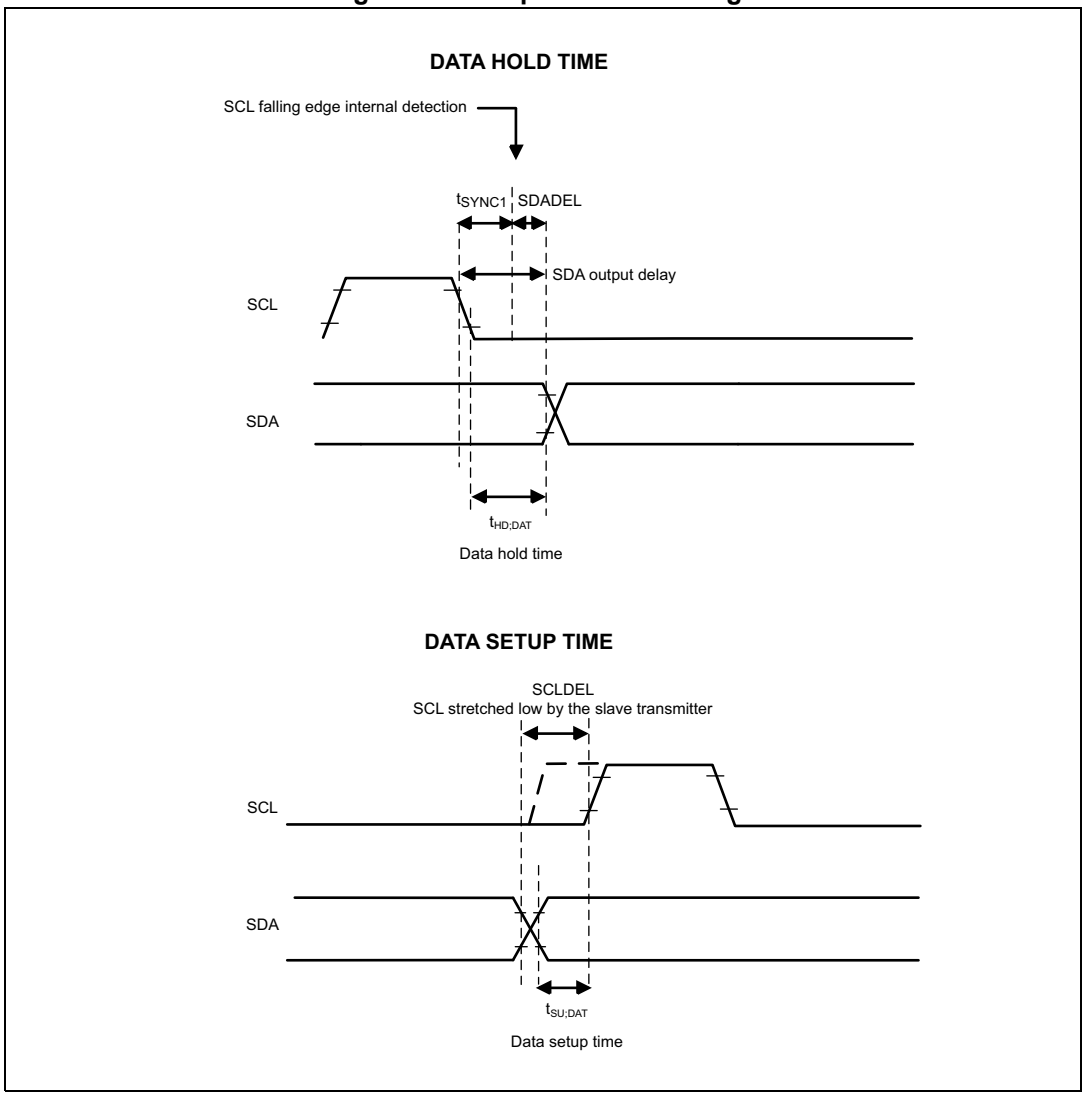
When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows to suppress spikes with a programmable length of 1 to 15 I2CCLK periods.

Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in controller and target modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

Figure 97. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.
  T_{SDADEL} impacts the hold time $t_{HD;DAT}$.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x t_{I2CCLK} }

t_{SYNC1} duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$ ns.
- When enabled, input delay brought by the digital filter: $t_{DNF} = DNF \times t_{I2CCLK}$
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, you must program SDADEL in such a way that:

$$\{t_{f\ (max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$$

$$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

Note: *$t_{AF(min)}$ / $t_{AF(max)}$ are part of the equation only when the analog filter is enabled. Refer to the datasheet [1] for t_{AF} values.*

The maximum $t_{HD;DAT}$ could be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}.$$

Note: *This condition can be violated when NOSTRETCH=0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.*

Refer to *Table 89: I2C-SMBUS specification data setup and hold times* for t_f, t_r, $t_{HD;DAT}$ and $t_{VD;DAT}$ standard values.

- After sending SDA output, SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), you must program SCLDEL in such a way that:

$$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 <= SCLDEL$$

Refer to *Table 89: I2C-SMBUS specification data setup and hold times* for t_r and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

Table 89. I²C-SMBUS specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100		50		250		ns
t_r	Rise time of both SDA and SCL signals	-	1000		300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300		300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is $t_{SCLL} = (SCLL+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$.
  t_{SCLL} impacts the SCL low time t_{LOW} .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is $t_{SCLH} = (SCLH+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLH} impacts the SCL high time t_{HIGH} .

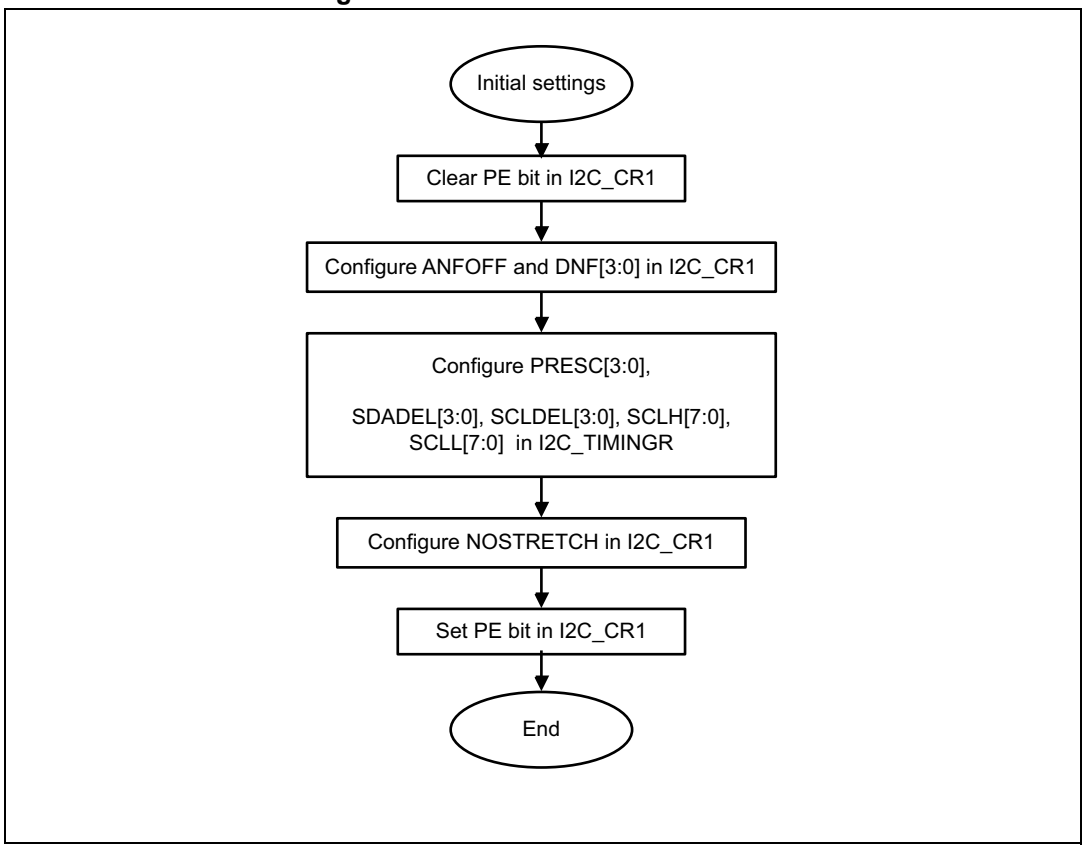
Refer to section *21.4.4: I2C initialization* for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C target NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *I2C target initialization* for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 98. I2C initialization flowchart



21.4.5 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.

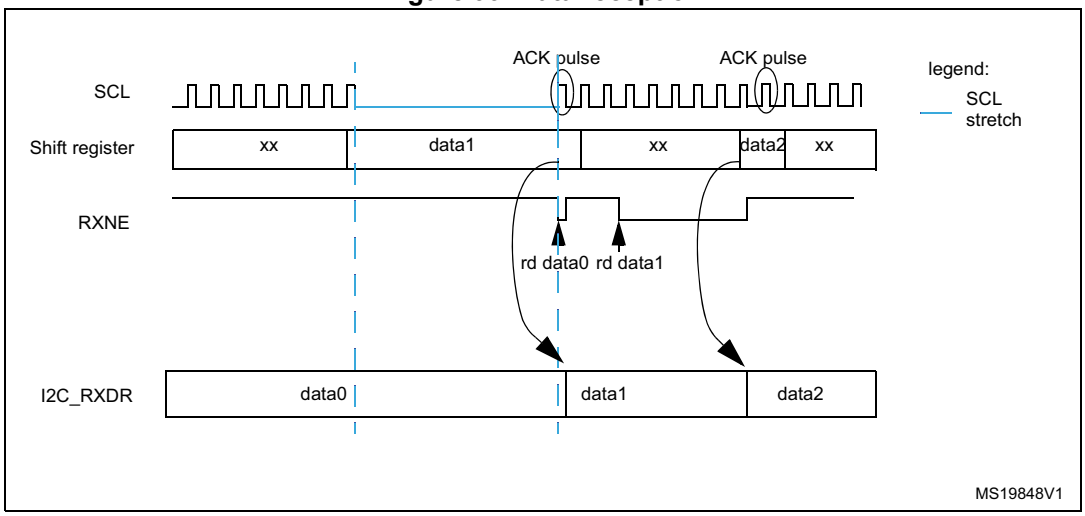
21.4.6 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

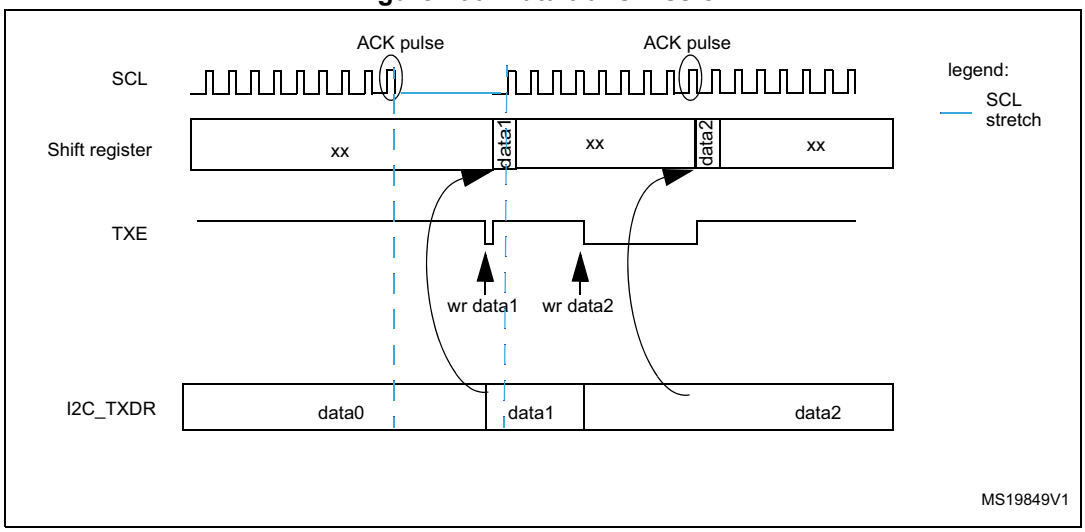
Figure 99. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 100. Data transmission



Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in controller mode. By default it is disabled in target mode, but it can be enabled by software by setting the SBC (Target Byte Control) bit in the I2C_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in controller mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = ‘1’ in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.
- **Software end mode** (AUTOEND = ‘0’ in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 90. I2C configuration table

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	x	0	1
Controller Tx/Rx + NBYTES + RESTART	x	0	0
Target Tx/Rx all received bytes ACKed	0	x	x
Target Rx with ACK control	1	1	x

21.4.7 I2C target mode

I2C target initialization

In order to work in target mode, you must enable at least one target address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the target own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional target addresses are required, you can configure the 2nd target address OA2. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the controller does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled you must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

Target clock stretching (NOSTRETCH = 0)

In default mode, the I2C target stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled target addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag

is cleared (TXE=1). This stretch is released when the data is written to the I2C_TXDR register.

- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Target Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}.

Target without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C target does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if you clear the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, you ensure that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Target byte control mode

In order to allow byte ACK control in target reception mode, Target Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in target reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. You can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

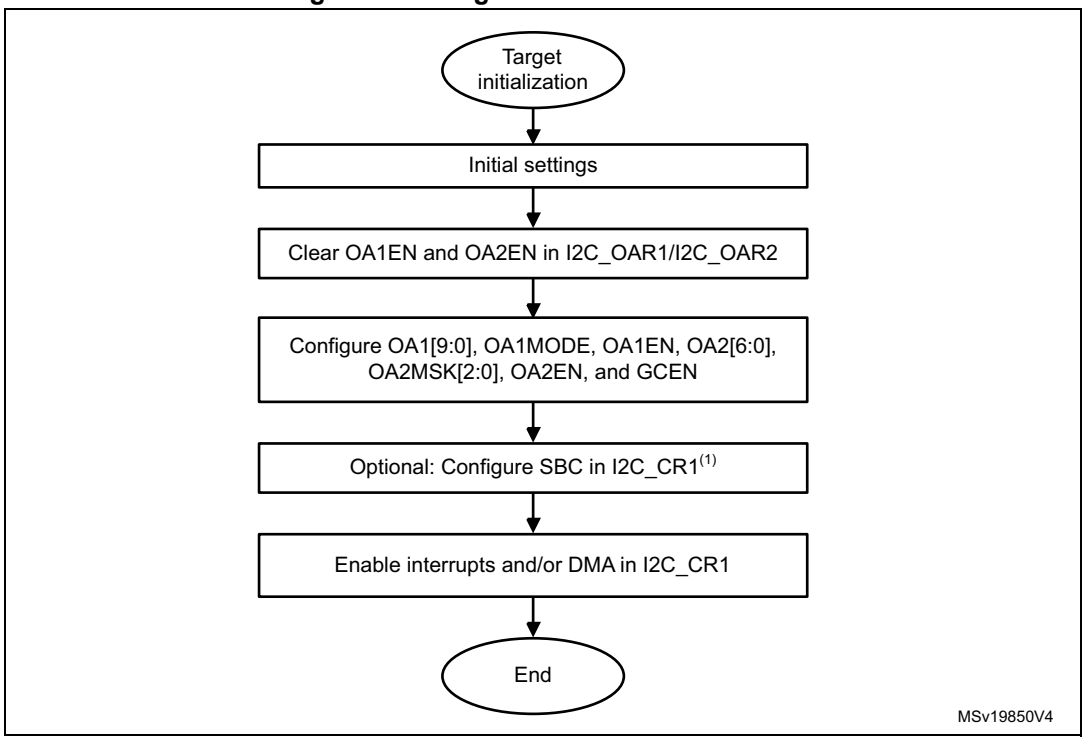
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the target is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Caution: Target Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

Figure 101. Target initialization flowchart



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The target automatically releases the SCL and SDA lines in order to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to ‘0’. In this case, If TXE = 0 when the target address is received (ADDR=1), you can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Target Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so you cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program

the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If you need a TXIS event, (Transmit Interrupt or Transmit DMA request), you must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 102. Transfer sequence flowchart for I2C target transmitter, NOSTRETCH=0

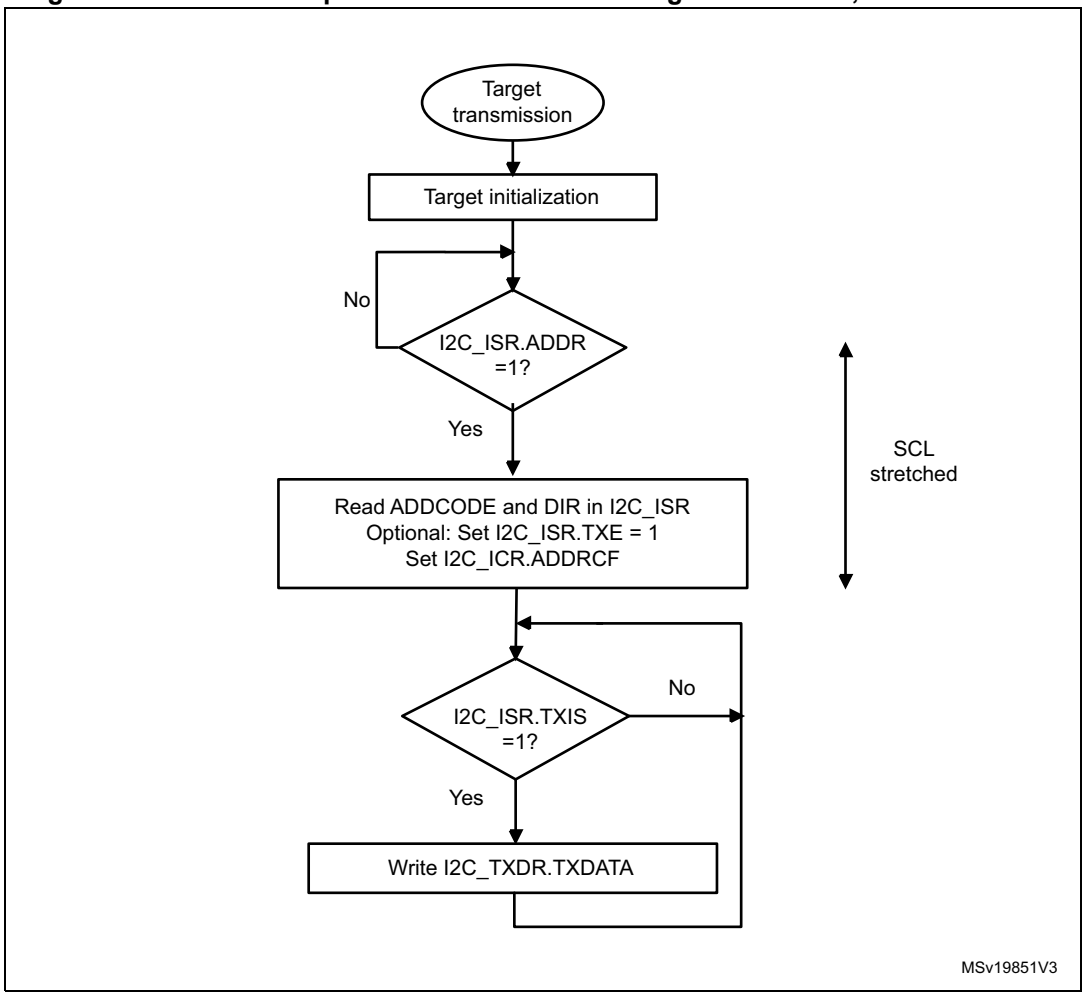


Figure 103. Transfer sequence flowchart for I2C target transmitter, NOSTRETCH=1

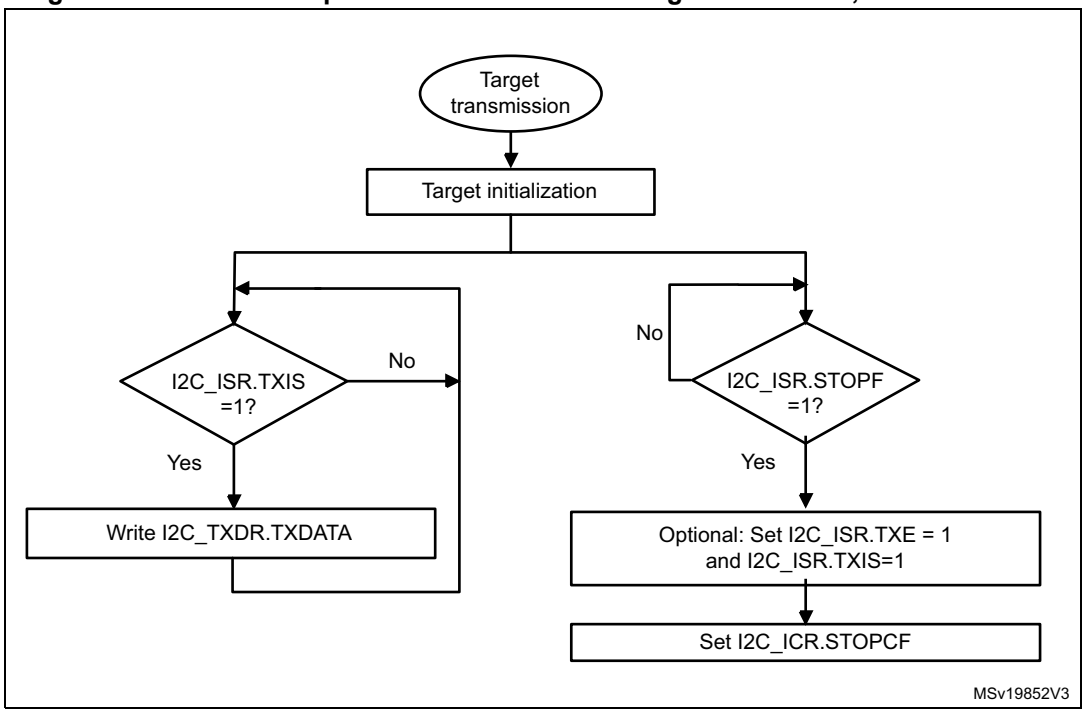
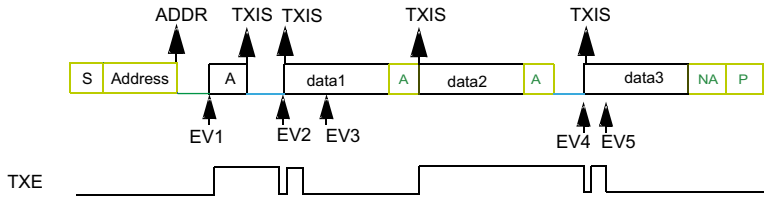


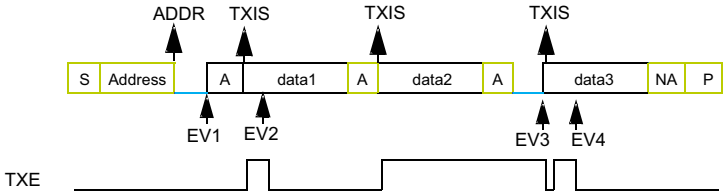
Figure 104. Transfer bus diagrams for I2C target transmitter

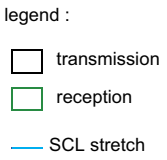
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

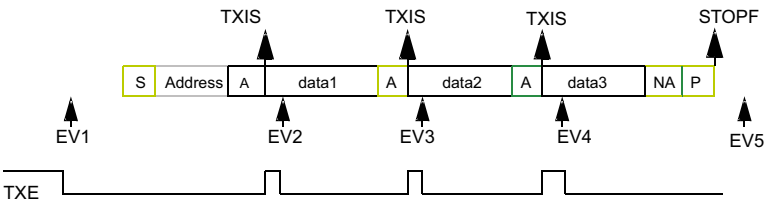
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:





EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 105. Transfer sequence flowchart for target receiver with NOSTRETCH=0

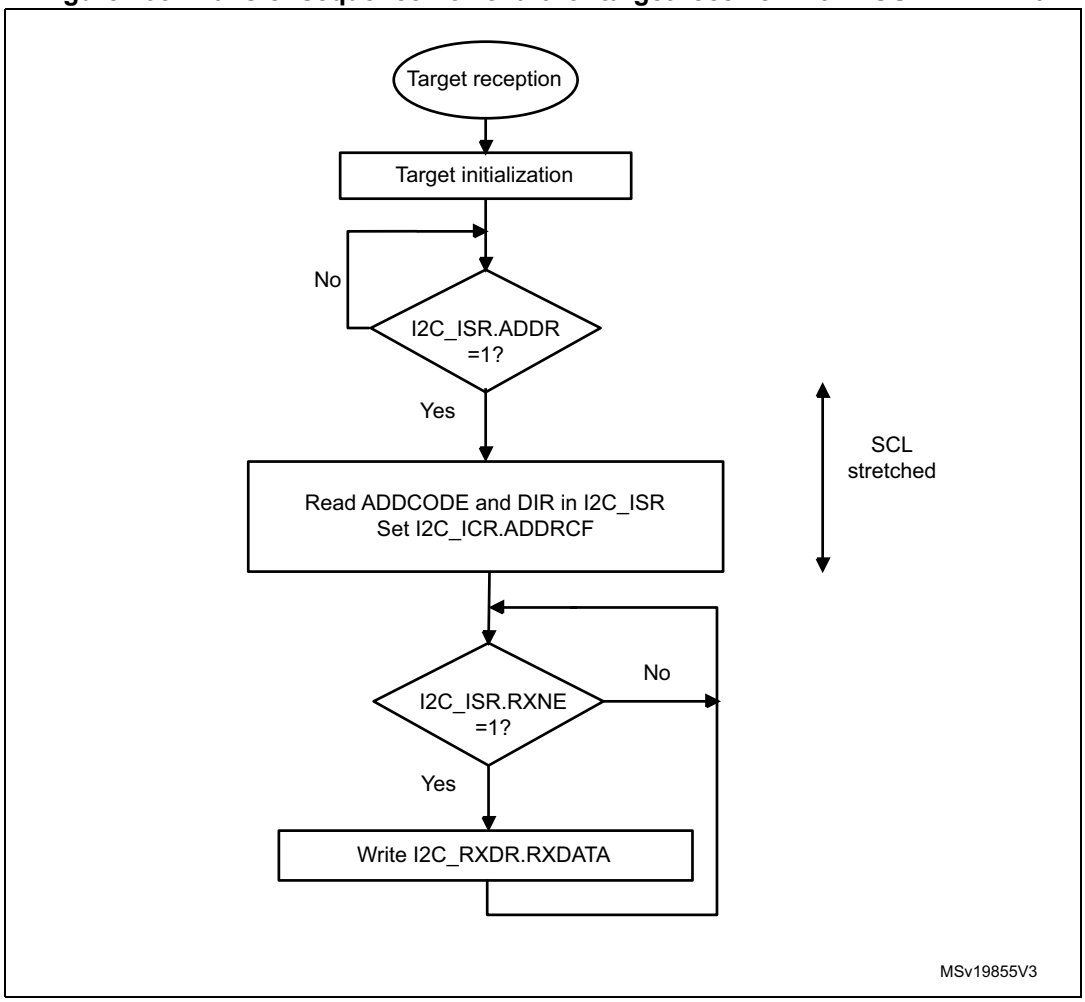


Figure 106. Transfer sequence flowchart for target receiver with NOSTRETCH=1

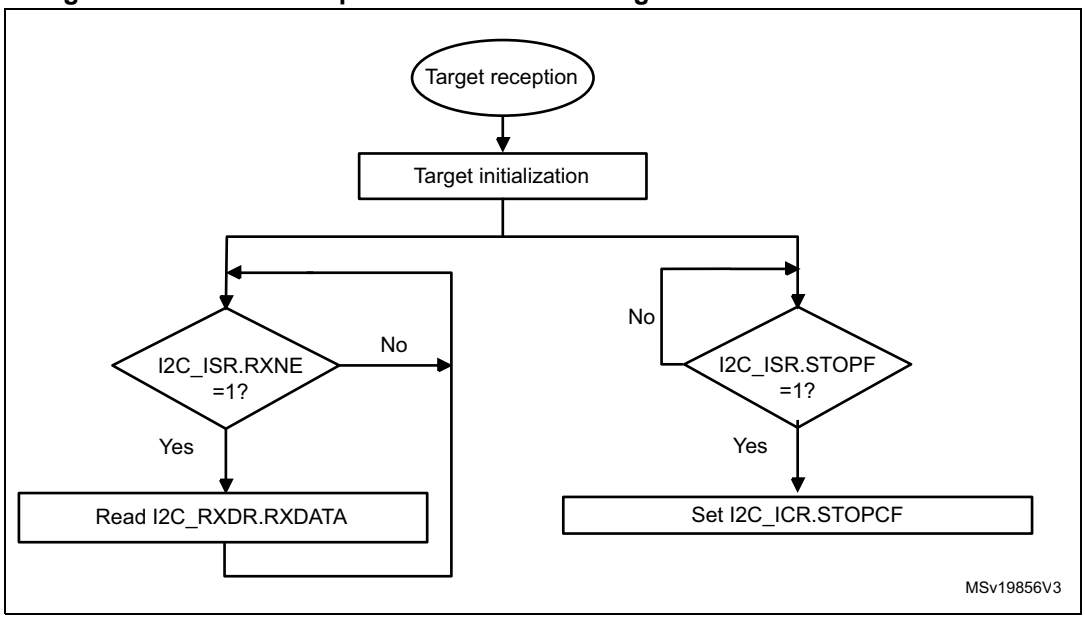
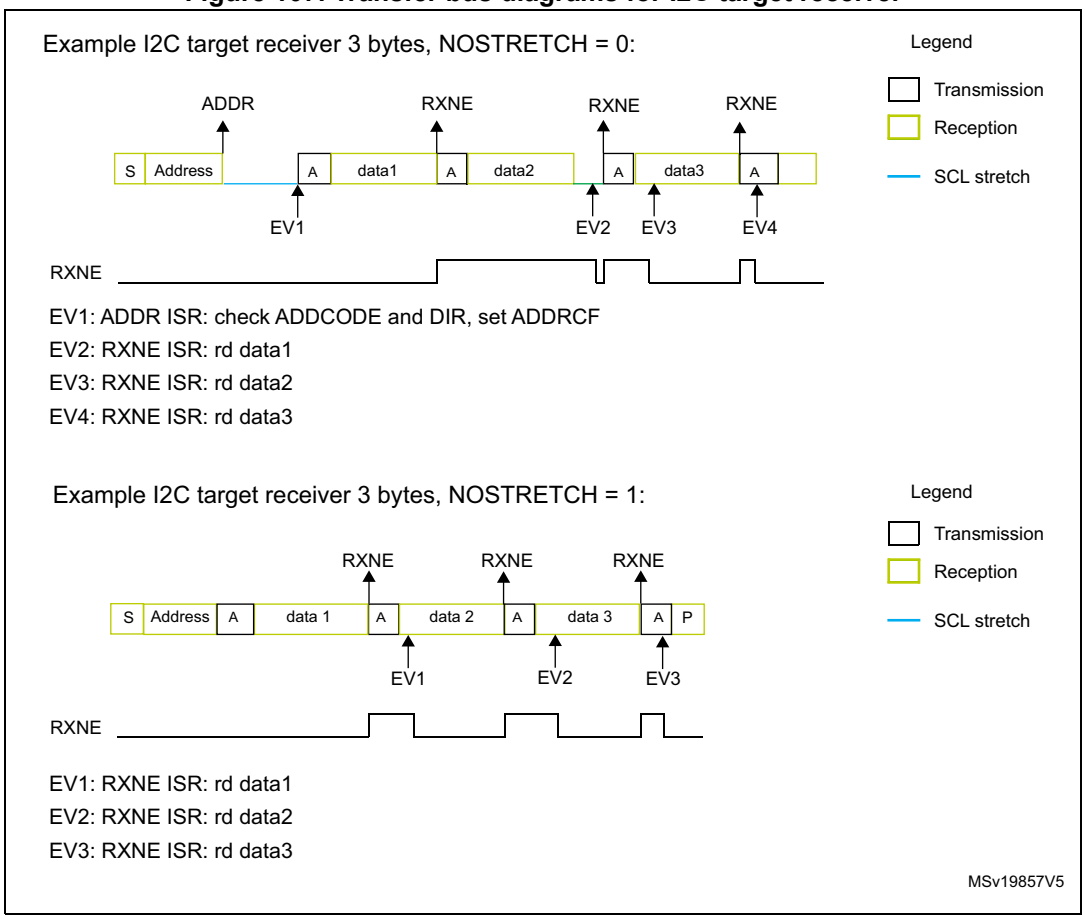


Figure 107. Transfer bus diagrams for I2C target receiver



21.4.8 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

A clock synchronization mechanism is implemented in order to support multi-controller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$

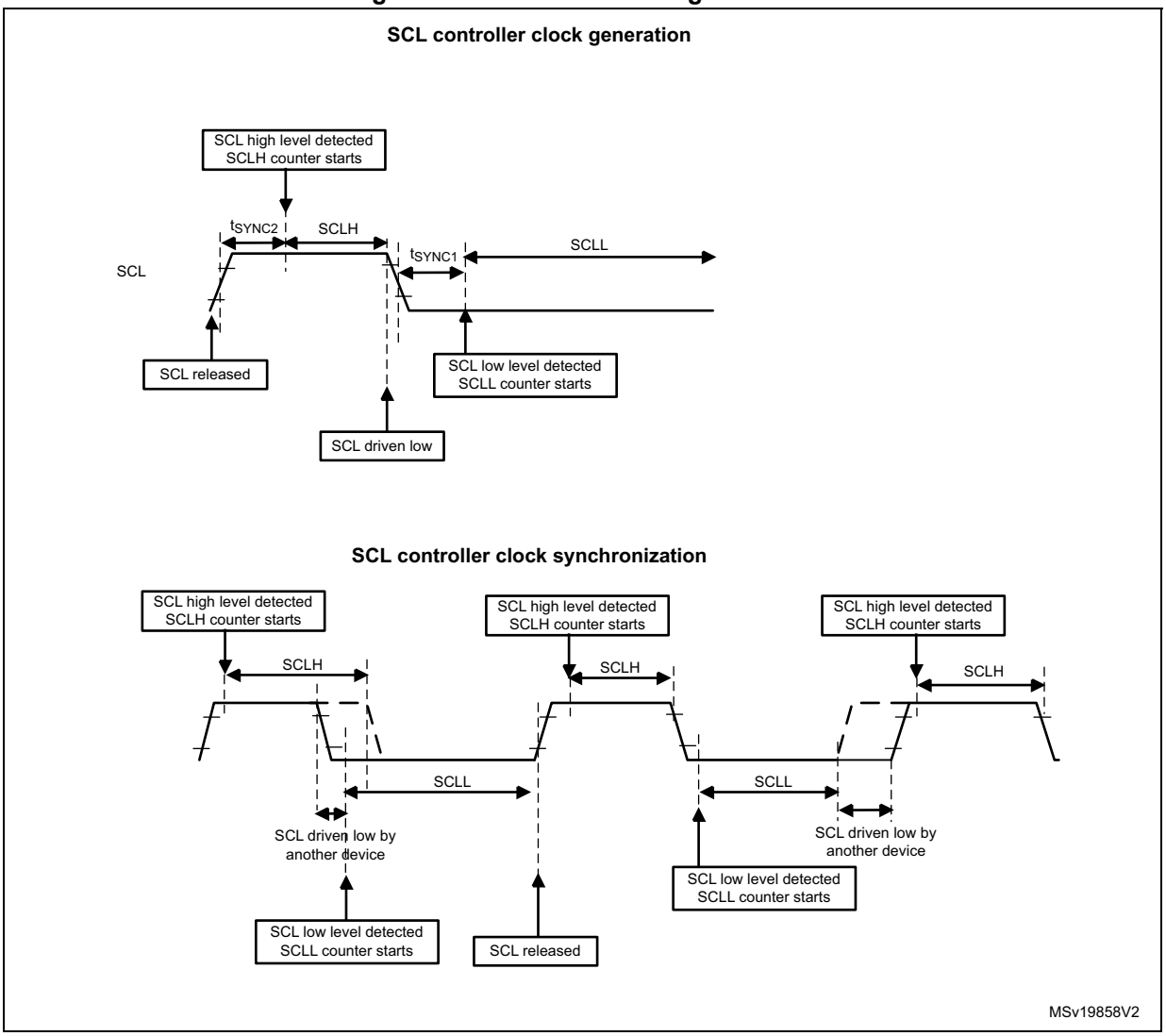
The duration of t_{SYNC1} depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of t_{SYNC2} depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

Figure 108. Controller clock generation



Caution: In order to be I^{2}C or SMBus compliant, the controller clock must respect the timings given below:

Table 91. I^{2}C-SMBUS specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100		400		1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6		0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6		0.26	-	4.7	-	µs
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6		0.26	-	4.0	-	µs

Table 91. I²C-SMBUS specification clock timings (continued)

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{LOW}	Low period of the SCL clock	4.7	-	1.3		0.5	-	4.7	-	µs
t_{HIGH}	Period of the SCL clock	4.0	-	0.6		0.26	-	4.0	50	µs
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	ns

Note: SCLL is also used to generate the t_{BUF} and $t_{SU:STA}$ timings.

SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 21.4.9: I2C_TIMINGR register configuration examples* for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

Controller communication initialization (address phase)

In order to initiate the communication, you must program the following parameters for the addressed target in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Target address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

You must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

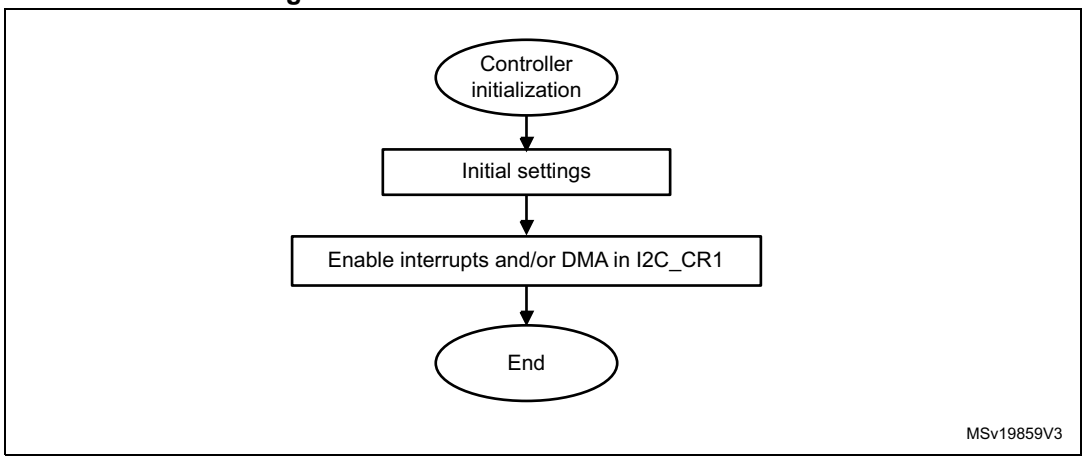
Then the controller automatically sends the START condition followed by the target address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF}.

In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs. If the I2C is addressed as a target (ADDR=1) while the START bit is set, the I2C switches to target mode and the START bit is cleared when the ADDRCF bit is set.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

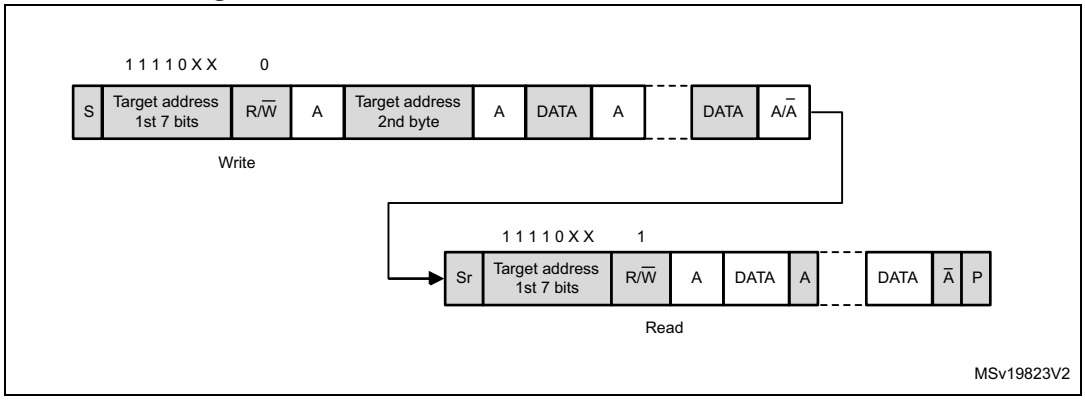
Figure 109. Controller initialization flowchart



Initialization of a controller receiver addressing a 10-bit address target

- If the target address is in 10-bit format, you can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the controller automatically sends the following complete sequence after the START bit is set: (Re)Start + Target address 10-bit header Write + Target address 2nd byte + REStart + Target address 10-bit header Read
- If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated start is set with the 10 bit target address configured with HEAD10R=1. In this case the controller sends this sequence: ReStart + Target address 10-bit header Read

Figure 110. 10-bit address read access with HEAD10R=1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when

NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper target address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 111. Transfer sequence flowchart for I2C controller transmitter for N≤255 bytes

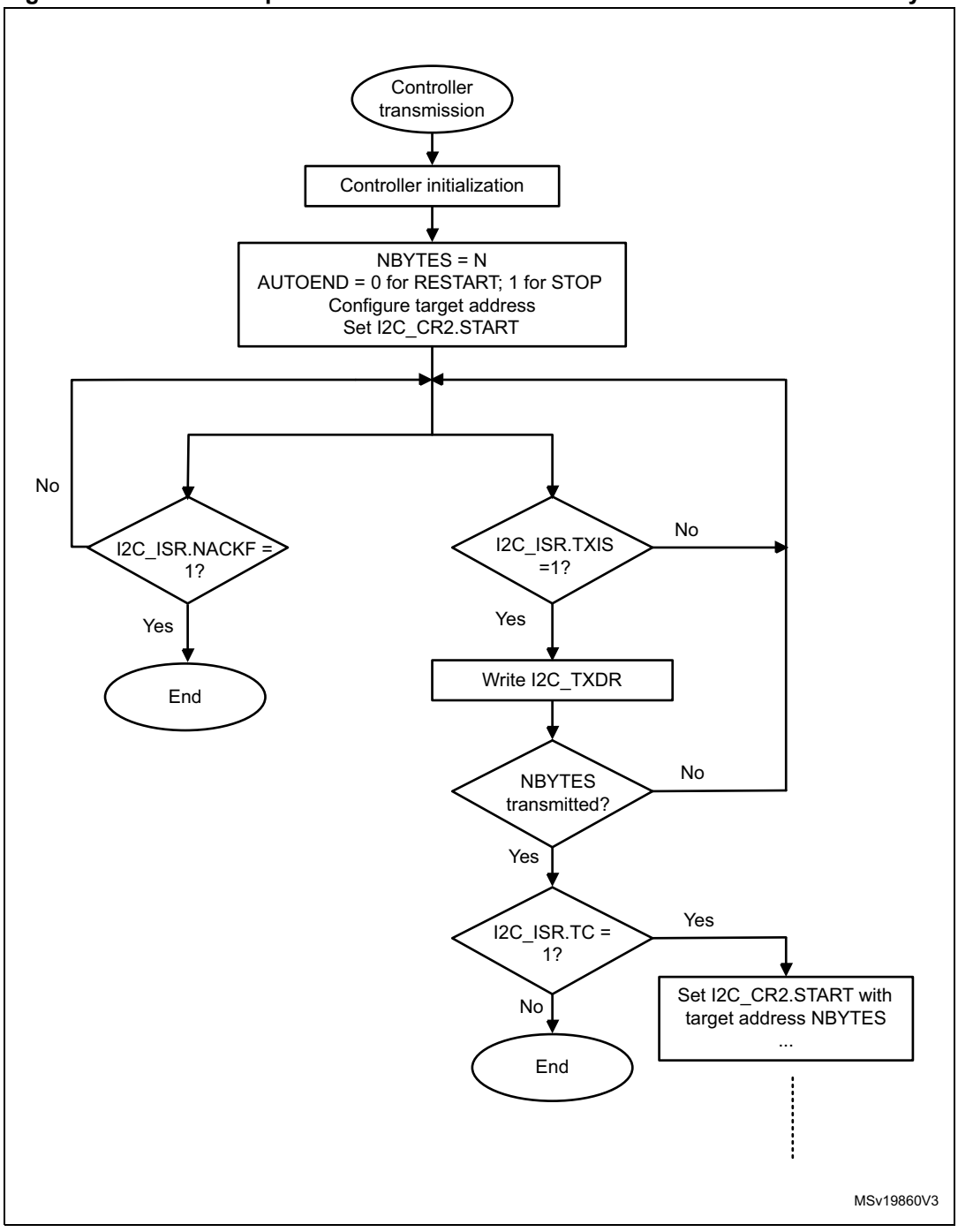


Figure 112. Transfer sequence flowchart for I2C controller transmitter for N>255 bytes

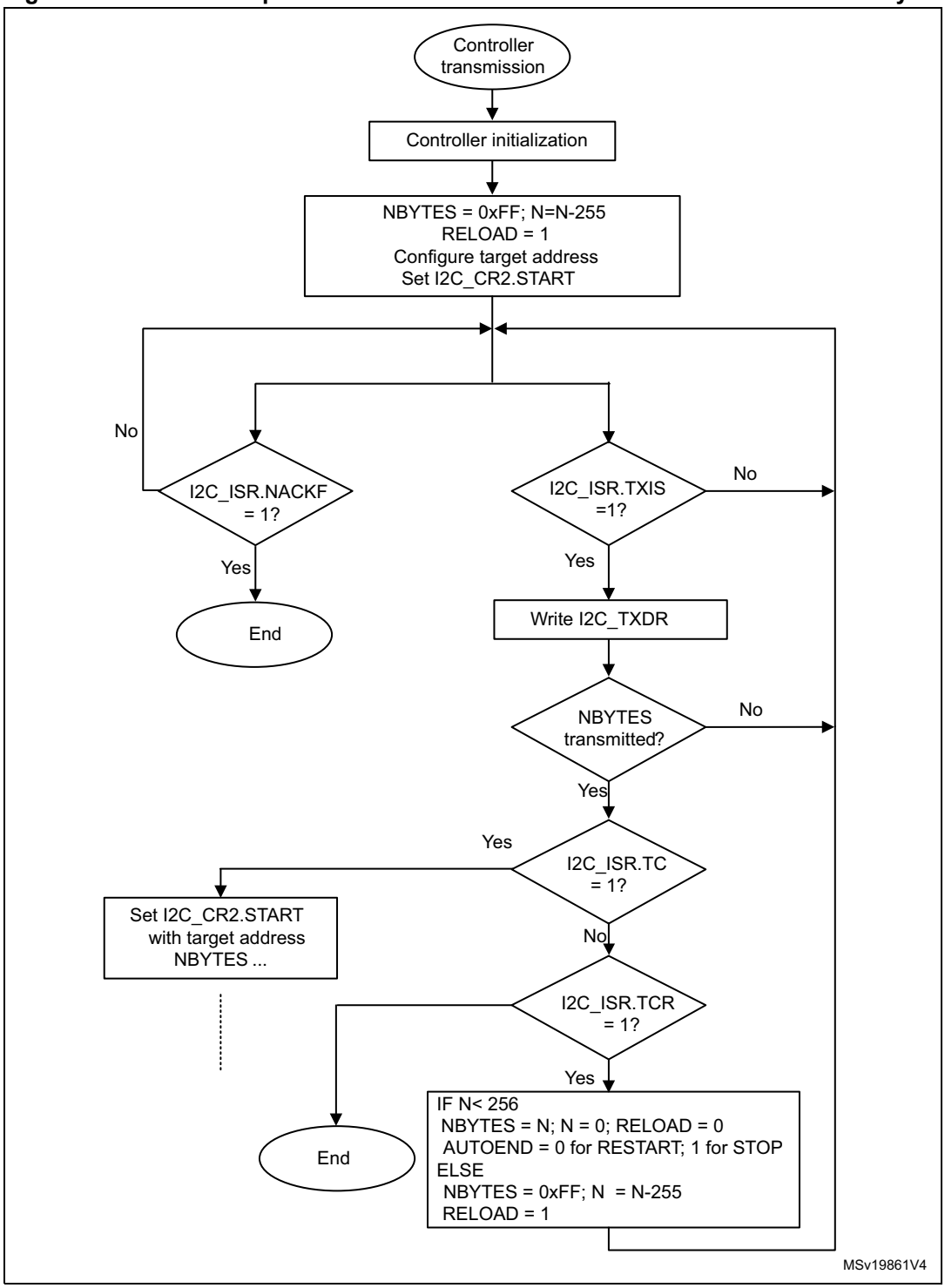
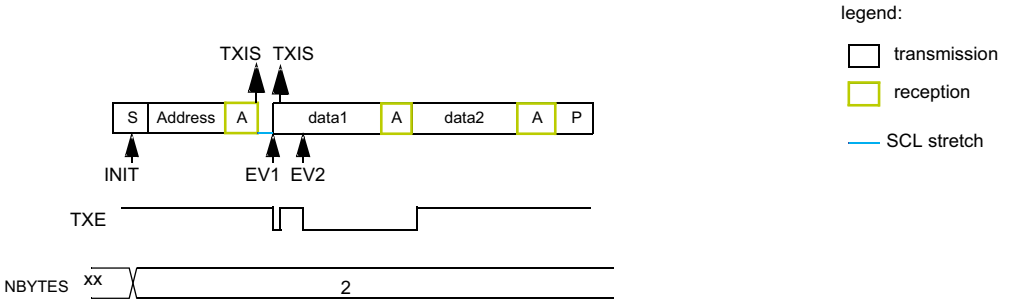


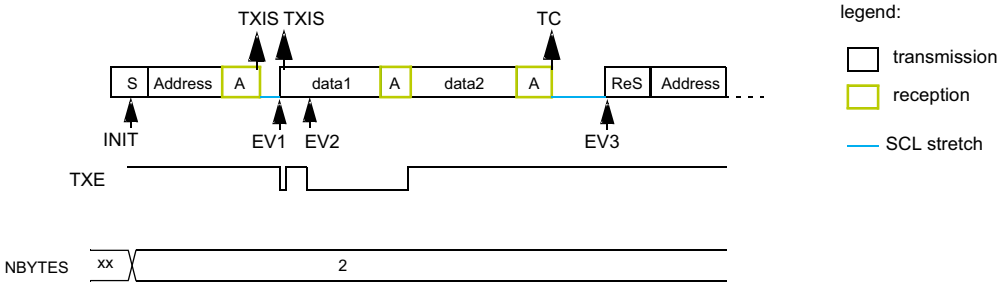
Figure 113. Transfer bus diagrams for I2C controller transmitter

Example I2C controller transmitter 2 bytes, automatic end mode (STOP)



INIT: program target address, program NBYTES = 2, AUTOEND = 1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C controller transmitter 2 bytes, software end mode (RESTART)



INIT: program target address, program NBYTES = 2, AUTOEND = 0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program target address, program NBYTES = N, set START

MSv19862V3

Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper target address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by target address, are sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 114. Transfer sequence flowchart for I2C controller receiver for N≤255 bytes

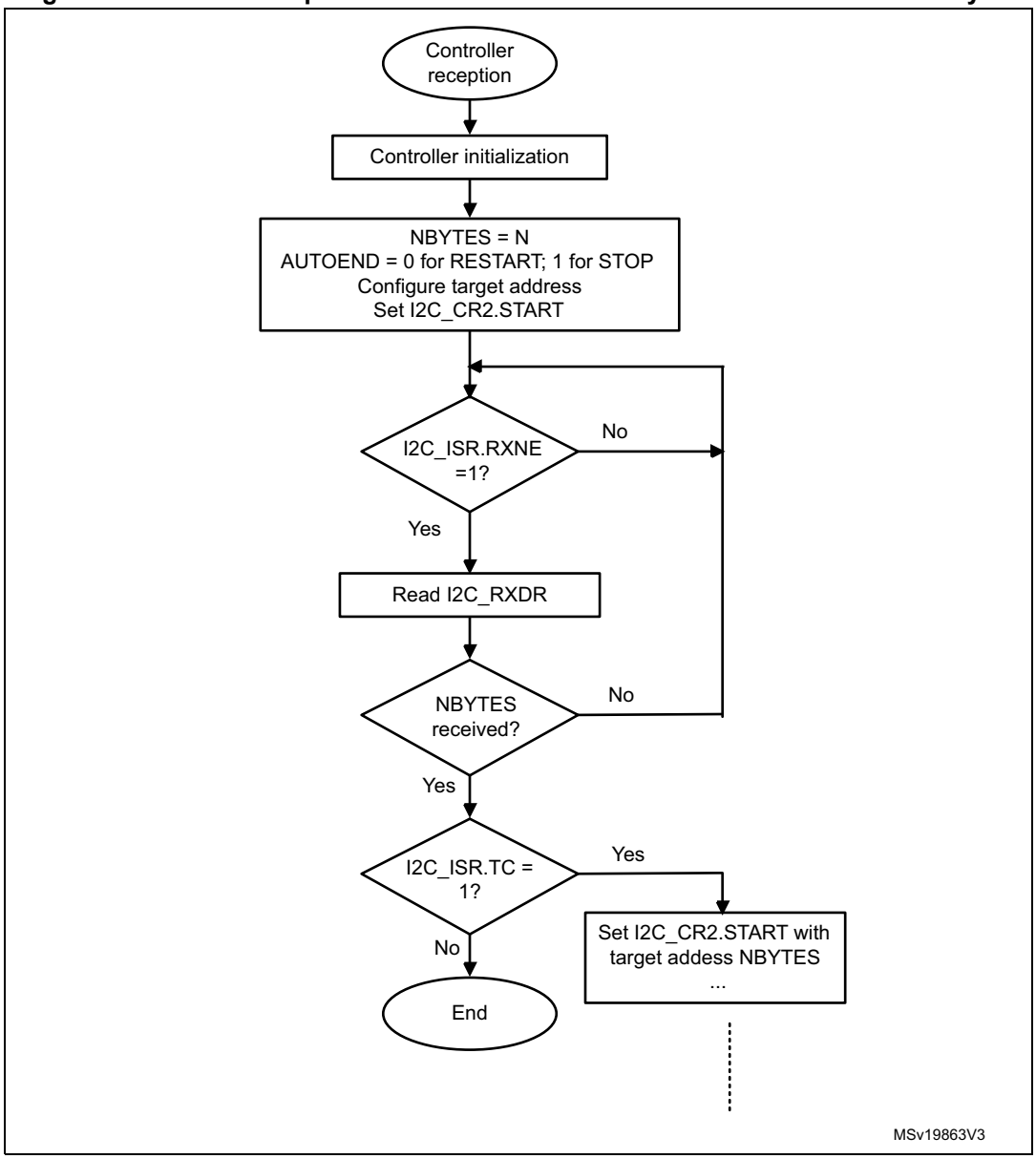


Figure 115. Transfer sequence flowchart for I2C controller receiver for N >255 bytes

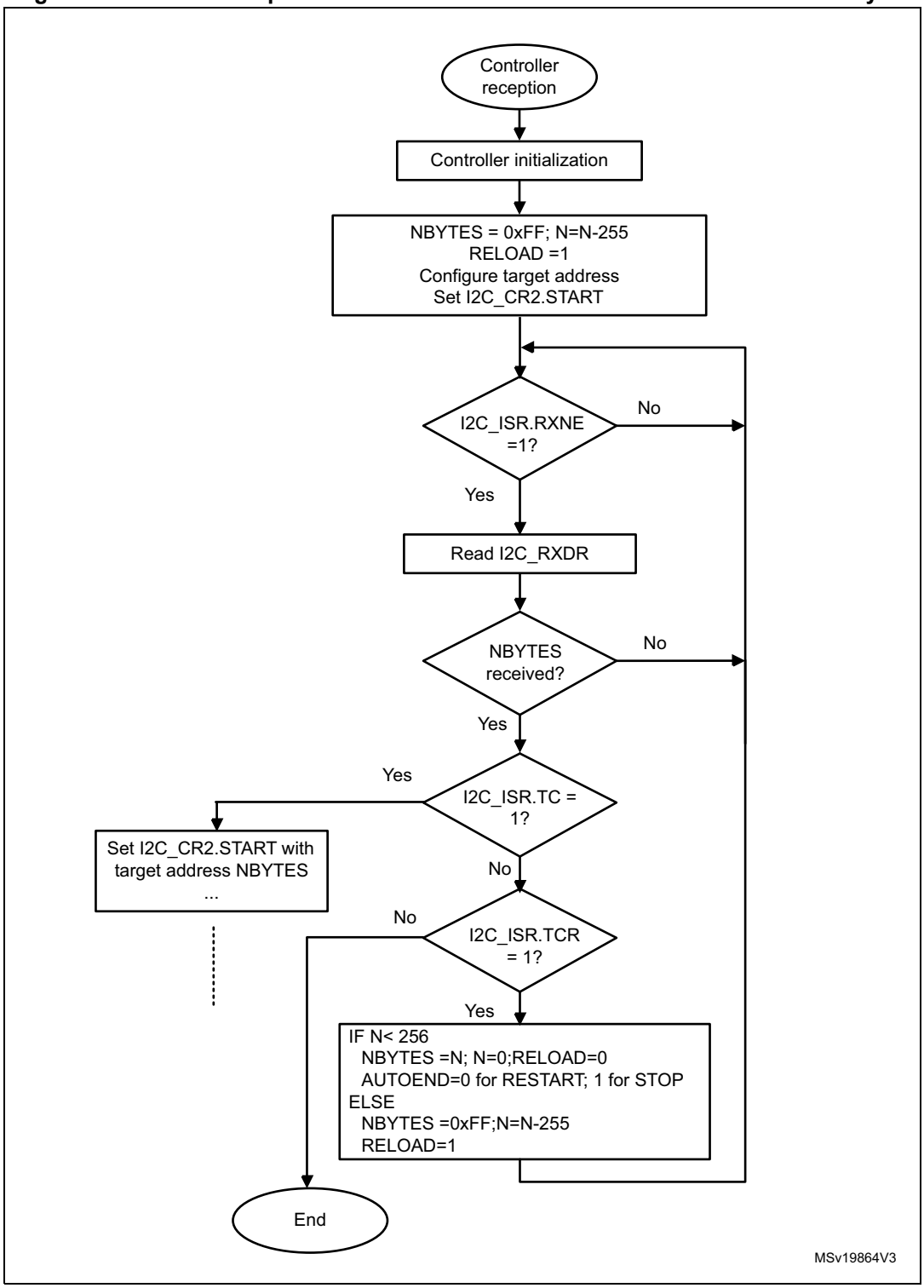
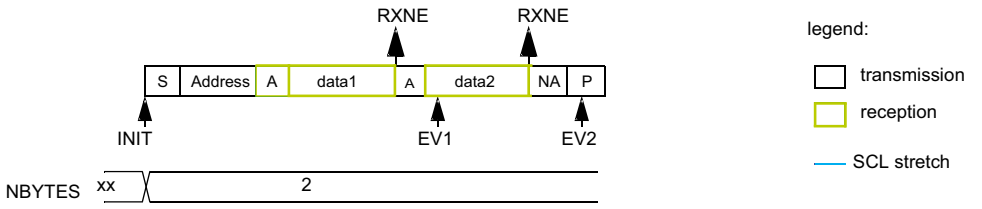


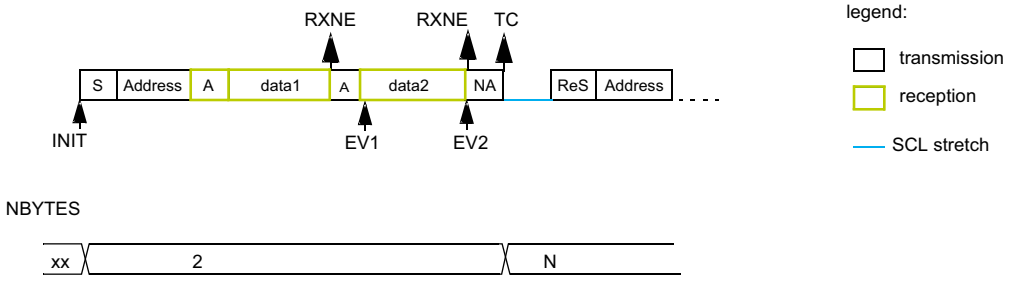
Figure 116. Transfer bus diagrams for I2C controller receiver

Example I2C controller receiver 2 bytes, automatic end mode (STOP)



INIT: program target address, program NBYTES = 2, AUTOEND=1, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2

Example I2C controller receiver 2 bytes, software end mode (RESTART)



INIT: program target address, program NBYTES = 2, AUTOEND=0, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: read data2
EV3: TC ISR: program target address, program NBYTES = N, set START

MSv19865V2

21.4.9 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I^2C specification.

Table 92. Examples of timings settings for f_{I2CCLK} = 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns

21.4.10 SMBus specific features

Table 93. Examples of timings settings for f_{I2CCLK} = 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~875 ns(4)
SDADEL	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. The SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns
3. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns
4. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 250 ns

This section is relevant only when SMBus feature is supported. Refer to *Section 21.3: I2C implementation*.

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification rev 2.0 (http://smbus.org).

The System Management Bus Specification refers to three types of devices.

- A target is a device that receives or responds to a command.
- A controller is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized controller that provides the main interface to the system's CPU. A host must be a controller-target and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as controller or target device, and also as a host.

SMBUS is based on I^2C specification rev 2.1.

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification version 2.0 (http://smbus.org).

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification version 2.0 (http://smbus.org).

Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the Target Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to *SMBus target mode on page 493* section for more details.

Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus Host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The SMBus ALERT optional signal is supported. A target-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the Alert Response Address.

When configured as a target device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is

generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0*.*

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x_8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

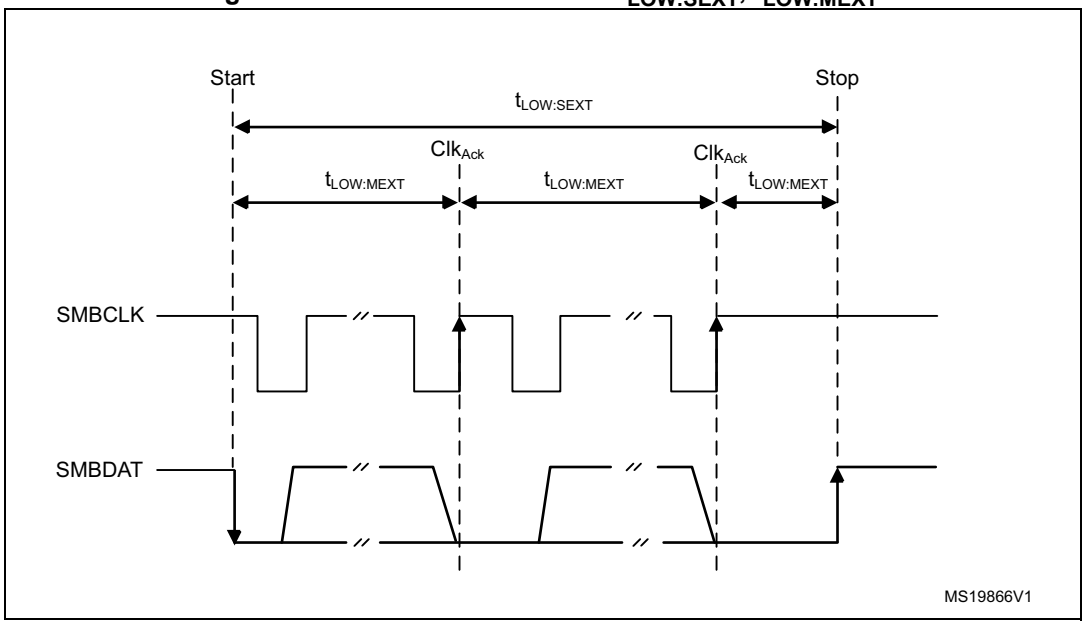
This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification version 2.0.

Table 94. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	ms
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	ms

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. Therefore, this parameter is measured with the target device as the sole target of a full-speed controller.
2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed target device as the sole target of the controller.

Figure 117. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{HIGH,MAX}$. (refer to *Table 91: I2C-SMBUS specification clock timings*)

This timing parameter covers the condition where a controller has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

21.4.11 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 21.3: I2C implementation*.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received Command and Data Acknowledge control (Target mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in target mode, the Target Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to *SMBus target mode on page 493* for more details.

Specific address (Target mode)

The specific SMBus addresses should be enabled if needed. Refer to *Bus idle detection on page 490* for more details.

- The SMBus Device Default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus Host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.

Table 95. SMBUS with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Target Tx/Rx with PEC	1	0	x	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification version 2.0.

- $t_{TIMEOUT}$ check

  In order to enable the $t_{TIMEOUT}$ check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the $t_{TIMEOUT}$ parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.

  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.

  If SCL is tied low for a time greater than (TIMEOUTA+1) x 2048 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 96: Examples of TIMEOUTA settings (max $t_{TIMEOUT}$ = 25 ms)*.

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

  Depending on if the peripheral is configured as a controller or as a target, The 12-bit TIMEOUTB timer must be configured in order to check $t_{LOW:SEXT}$ for a target and

$t_{LOW:MEXT}$ for a controller. As the standard specifies only a maximum, you can choose the same value for the both.

Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB+1) x 2048 x t_{I2CCLK}, and in the timeout interval described in *Bus idle detection on page 490* section, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 97: Example of TIMEOUTB settings*

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

Bus Idle detection

In order to enable the t_{IDLE} check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the t_{IDLE} parameter. The TIDLE bit must be configured to ‘1 in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than (TIMEOUTA+1) x 4 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 98: Example of TIMEOUTA settings (max t_{IDLE} = 50 µs)*

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

21.4.12 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 21.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 96. Examples of TIMEOUTA settings (max $t_{TIMEOUT}$ = 25 ms)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 97. Example of TIMEOUTB settings

f_{I2CCLK}	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs

Table 98. Example of TIMEOUTA settings (max t_{IDLE} = 50 µs)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	t_{TIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

21.4.13 SMBus target mode

This section is relevant only when SMBus feature is supported. Refer to *Section 21.3: I2C implementation*.

In addition to 2C target transfer management (refer to *Section 21.4.7: I2C target mode*) some additional software flowcharts are provided to support SMBus.

SMBus Target transmitter

When the IP is used in SMBus, SBC must be programmed to ‘1’ in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES-1 and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the NBYTES-1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 118. Transfer sequence flowchart for SMBus target transmitter N bytes + PEC

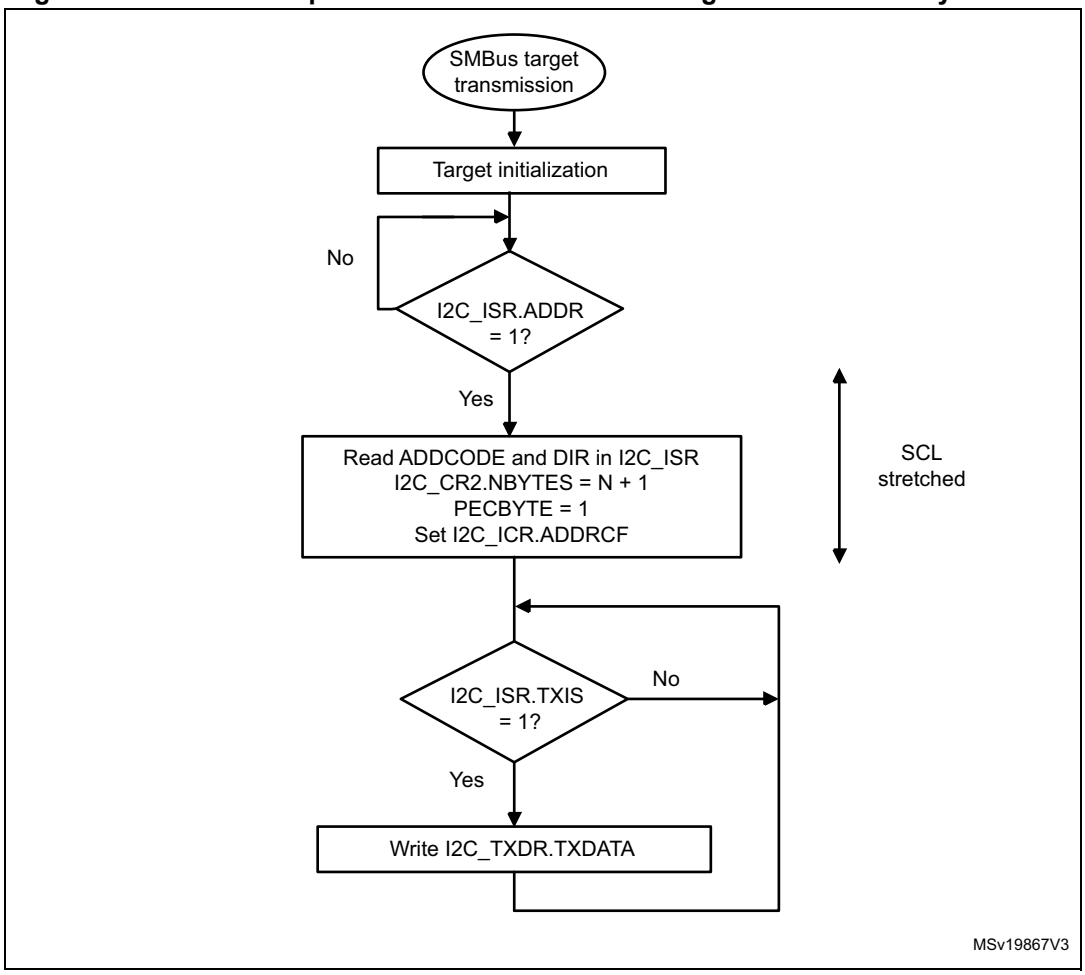
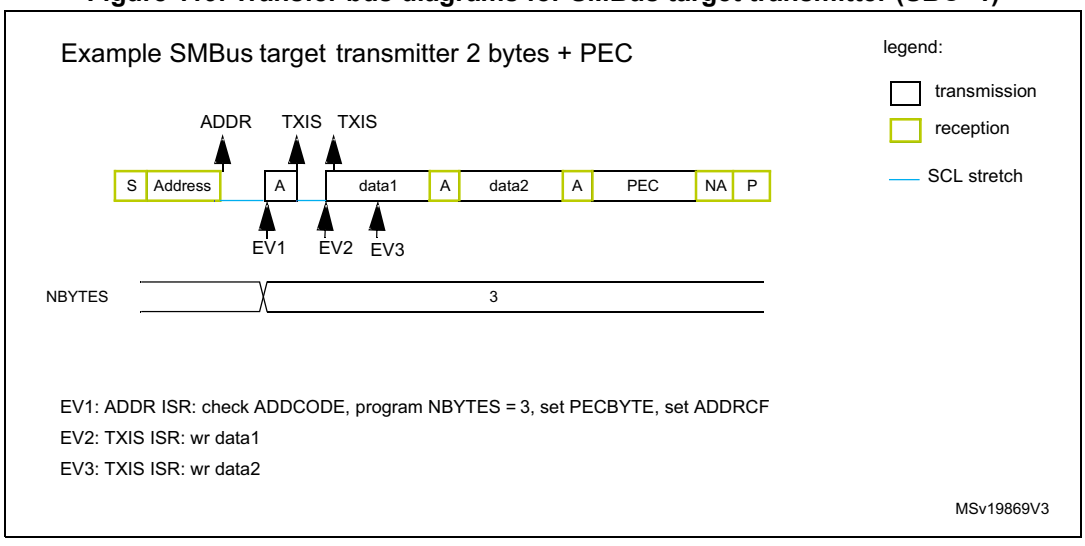


Figure 119. Transfer bus diagrams for SMBus target transmitter (SBC=1)



SMBus Target receiver

When the I2C is used in SMBus mode, SBC must be programmed to ‘1’ in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to *Target byte control mode on page 467* for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, you can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 120. Transfer sequence flowchart for SMBus target receiver N Bytes + PEC

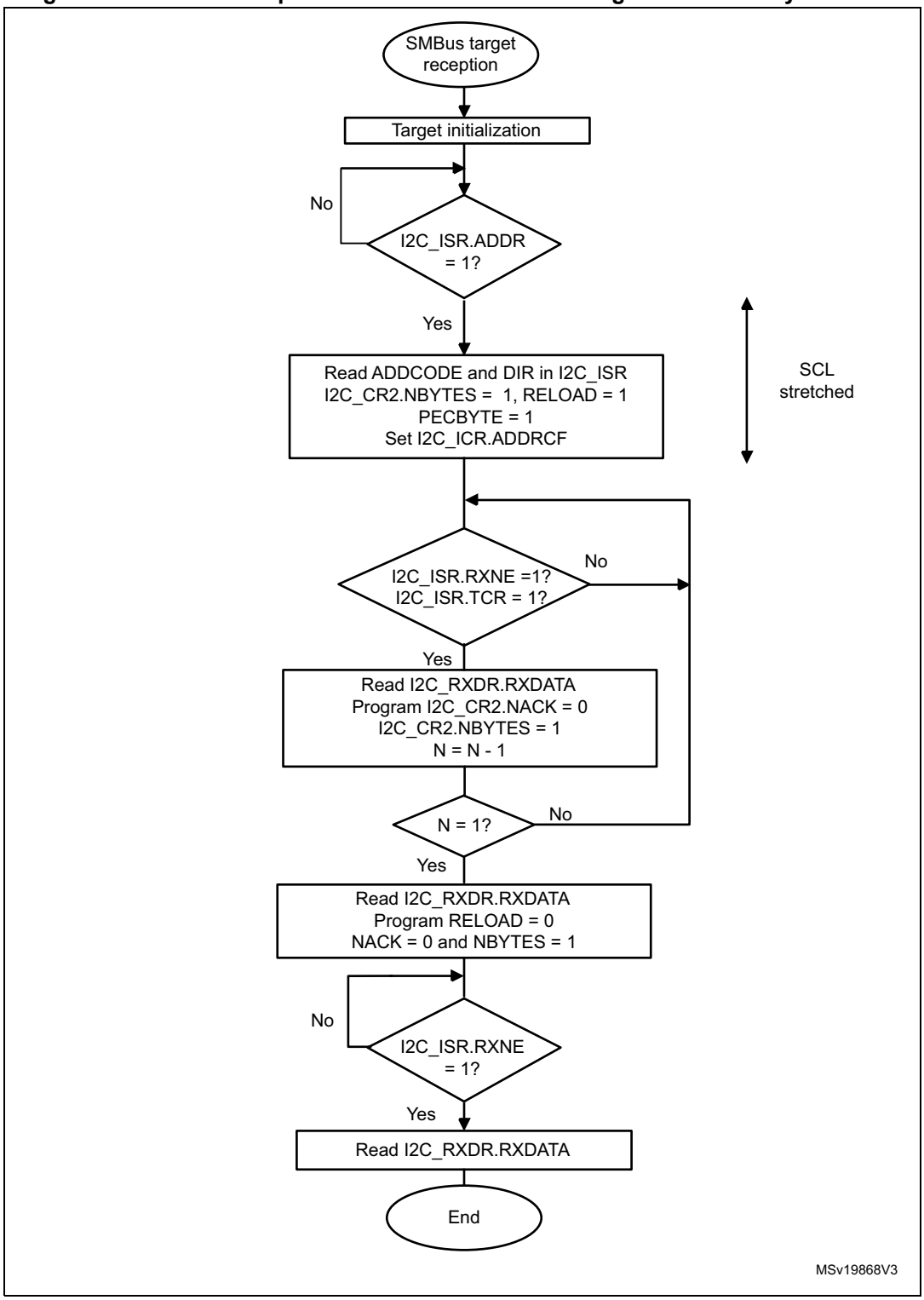
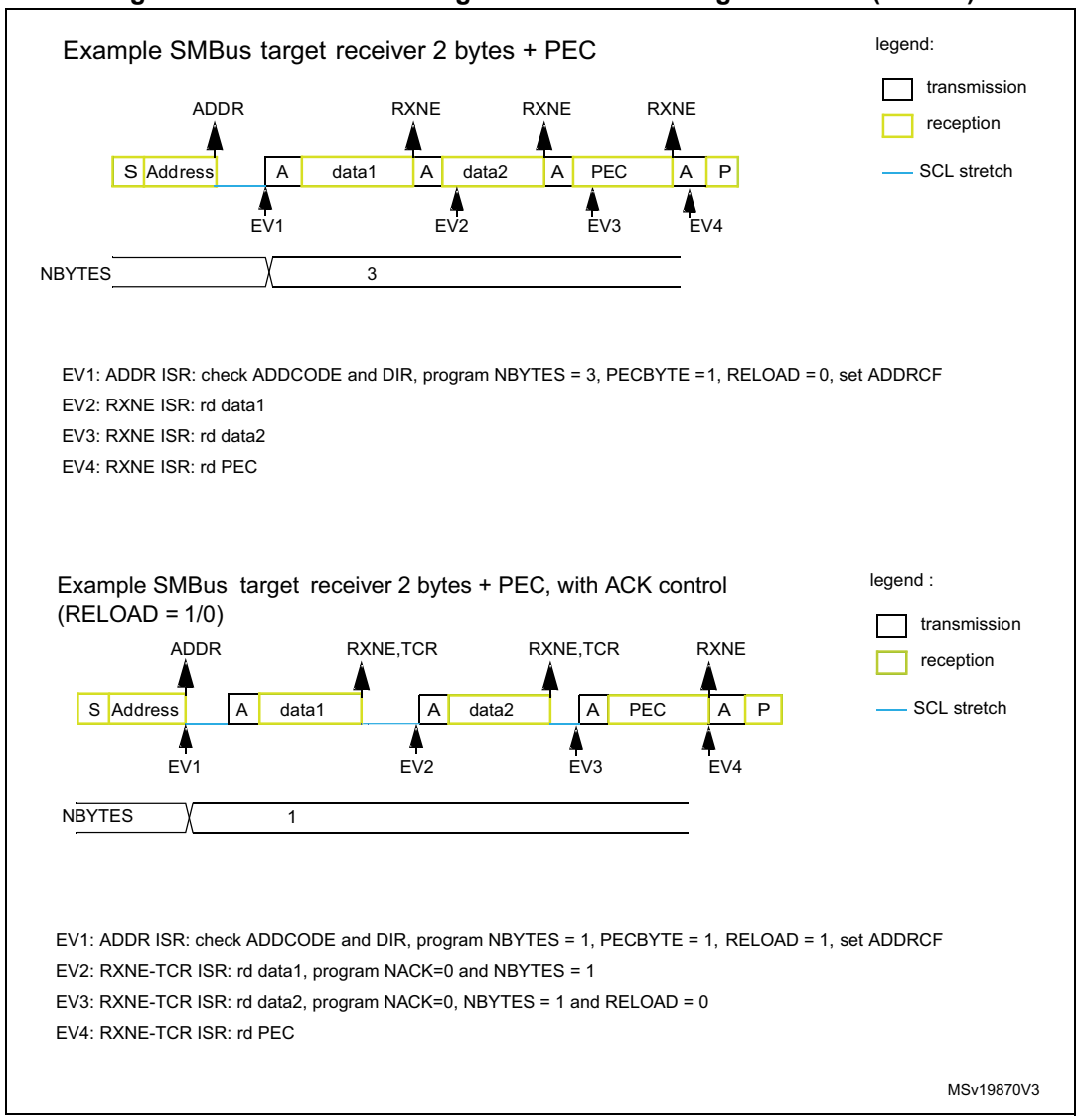


Figure 121. Bus transfer diagrams for SMBus target receiver (SBC=1)



This section is relevant only when SMBus feature is supported. Refer to *Section 21.3: I2C implementation*.

In addition to I2C controller transfer management (refer to *Section 21.4.8: I2C controller mode*) some additional software flowcharts are provided to support SMBus.

SMBus Controller transmitter

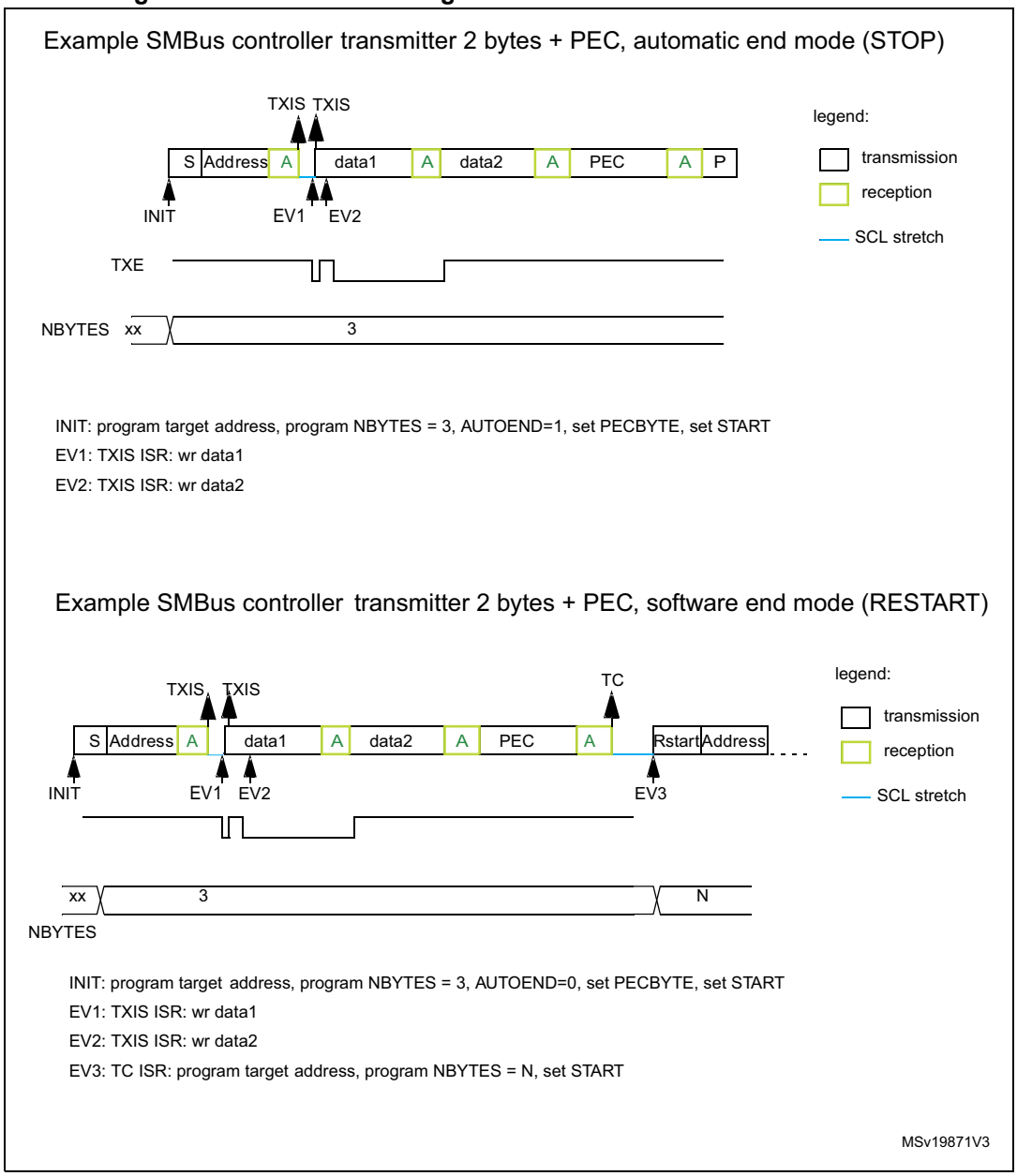
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, automatic end mode should be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 122. Bus transfer diagrams for SMBus controller transmitter



SMBus Controller receiver

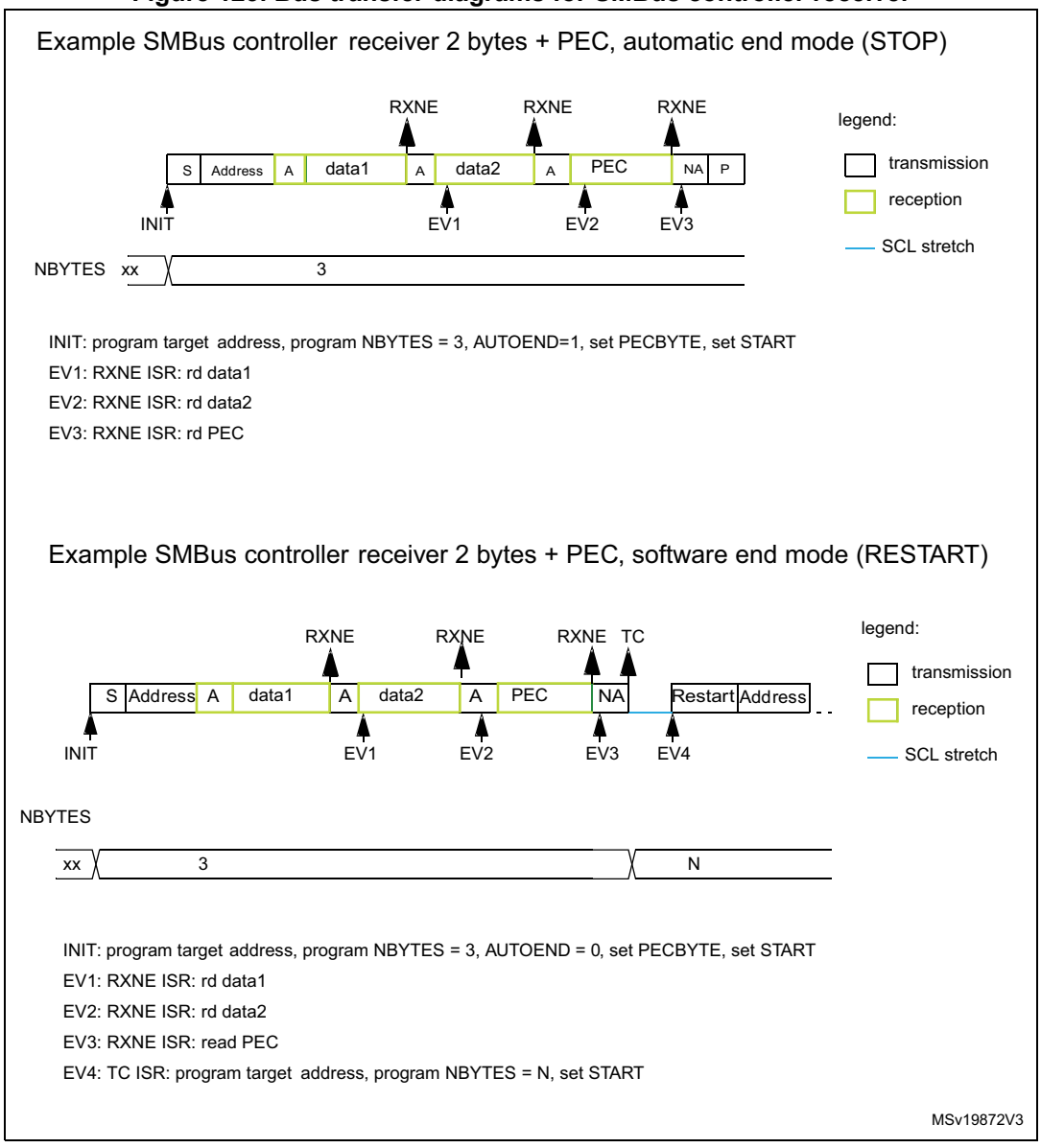
When the SMBus controller wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be

set and the target address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the target address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 123. Bus transfer diagrams for SMBus controller receiver



21.4.14 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as controller or addressed target (i.e not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.
- In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte should be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to *Section 21.3: I2C implementation*.

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 21.3: I2C implementation*.

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter)
- Target cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{LOW:SEXT}$ parameter)

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 21.3: I2C implementation*.

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

21.4.15 DMA requests

Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see *Section 10: DMA controller (DMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In controller mode: the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be

initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to Controller transmitter.

- In target mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to SMBus Target transmitter and SMBus Controller transmitter.

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to *Section 10: DMA controller (DMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.
- In target mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see *Section 21.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to SMBus Target receiver and SMBus Controller receiver.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

21.5 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 99. I2C Interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Receive buffer not empty	RXNE	Read I2C_RXDR register	RXIE
Transmit buffer interrupt status	TXIS	Write I2C_TXDR register	TXIE
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE
Transfer Complete Reload	TCR	Write I2C_CR2 with NBYTES[7:0] ≠ 0	TCIE
Transfer complete	TC	Write START=1 or STOP=1	

Table 99. I2C Interrupt requests (continued)

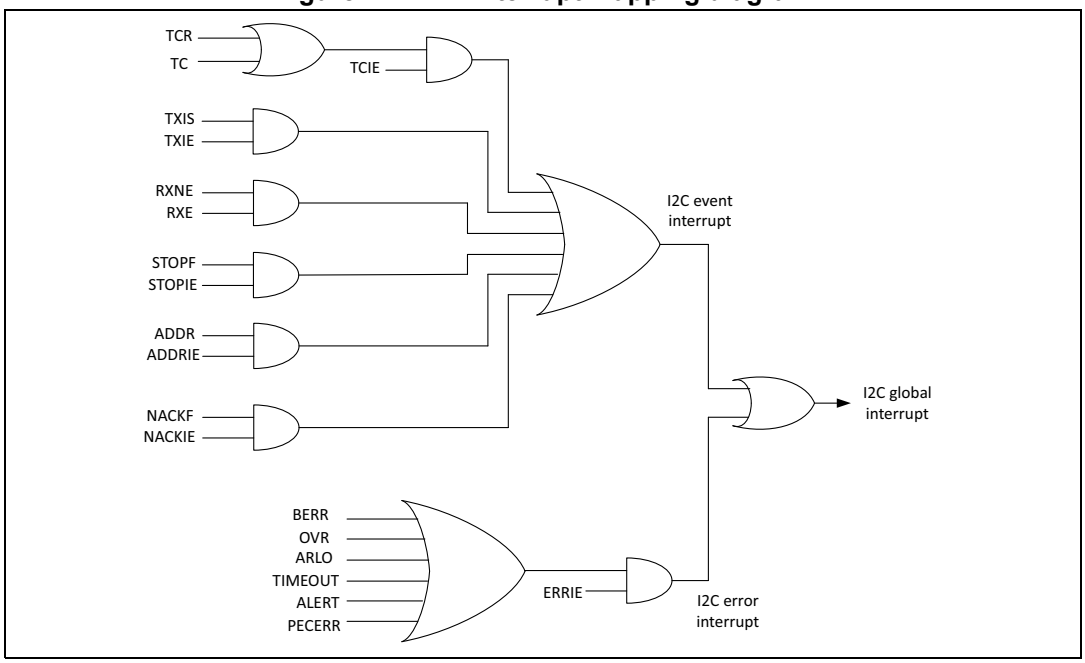
Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Address matched	ADDR	Write ADDRCF=1	ADDRIE
NACK reception	NACKF	Write NACKCF=1	NACKIE
Bus error	BERR	Write BERRCF=1	ERRIE
Arbitration loss	ARLO	Write ARLOCF=1	
Overrun/Underrun	OVR	Write OVRCF=1	
PEC error	PECERR	Write PECERRCF=1	
Timeout/t_{LOW} error	TIMEOUT	Write TIMEOUTCF=1	
SMBus Alert	ALERT	Write ALERTCF=1	

Depending on the product implementation, all these interrupts events can either share the same interrupt vector (I2C global interrupt), or be grouped into 2 interrupt vectors (I2C event interrupt and I2C error interrupt). Refer to *Section 2.3.2: Interrupts* for details.

In order to enable the I2C interrupts, the following sequence is required:

1. Configure and enable the I2C IRQ channel in the NVIC.
2. Configure the I2C to generate interrupts.

Figure 124. I2C interrupt mapping diagram



21.6 I2C registers

Refer to *Section 1.5: Acronyms* for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

21.6.1 Control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBD EN	SMBH EN	GCEN	Res.	NOSTR ETCH	SBC
								rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **PECEN:** PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 22 **ALERTEN**: SMBus alert enable

Device mode (SMBHEN=0):

0: Releases SMBA pin high and Alert Response Address Header disabled: 0001100x followed by NACK.

1: Drives SMBA pin low and Alert Response Address Header enables: 0001100x followed by ACK.

Host mode (SMBHEN=1):

0: SMBus Alert pin (SMBA) not supported.

1: SMBus Alert pin (SMBA) supported.

Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 21 **SMBDEN**: SMBus Device Default address enable

0: Device default address disabled. Address 0b1100001x is NACKed.

1: Device default address enabled. Address 0b1100001x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 20 **SMBHEN**: SMBus Host address enable
0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.
Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 19 **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
0: Clock stretching enabled
1: Clock stretching disabled
Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16 **SBC**: Target byte control
This bit is used to enable hardware byte control in target mode.
0: Target byte control disabled
1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter filter spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to15 t_{I2CCLK}
Note: If the analog filter is also enabled, the digital filter is added to the analog filter. This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generate an interrupt:
Arbitration Loss (ARLO)
Bus Error detection (BERR)
Overrun/Underrun (OVR)
Timeout detection (TIMEOUT)
PEC error detection (PECERR)
Alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer Complete interrupt enable
0: Transfer Complete interrupt disabled
1: Transfer Complete interrupt enabled
Note: Any of these events generates an interrupt:
Transfer Complete (TC)
Transfer Complete Reload (TCR)

Bit 5 **STOPIE**: STOP detection Interrupt enable
0: Stop detection (STOPF) interrupt disabled
1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disable
1: Peripheral enable
Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

21.6.2 Control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD 10R	ADD10	RD_W RN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw									

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE=0.

0: No PEC transfer.

1: PEC transmission/reception is requested

Note: Writing '0' to this bit has no effect.

This bit has no effect when RELOAD is set.

This bit has no effect is target mode when SBC=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES are reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing '0' to this bit has no effect.

This bit is used in target mode only: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (controller mode)

The bit is set by software, cleared by hardware when a Stop condition is detected, or when PE = 0.

In Controller Mode:

0: No Stop generation.

1: Stop generation after current byte transfer.

Note: Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCF bit in the I2C_ICR register.

0: No Start generation.

1: Restart/Start generation:

– If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.

– Otherwise setting this bit generates a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10 bit target address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

1: The controller only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode,

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer.

1: Controller requests a read transfer.

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:8 **SADD[9:8]**: Target address bit 9:8 (controller mode)
In 7-bit addressing mode (ADD10 = 0):
These bits are don't care
In 10-bit addressing mode (ADD10 = 1):
These bits should be written with bits 9:8 of the target address to be sent
Note: Changing these bits when the START bit is set is not allowed.

Bits 7:1 **SADD[7:1]**: Target address bit 7:1 (controller mode)
In 7-bit addressing mode (ADD10 = 0):
These bits should be written with the 7-bit target address to be sent
In 10-bit addressing mode (ADD10 = 1):
These bits should be written with bits 7:1 of the target address to be sent.
Note: Changing these bits when the START bit is set is not allowed.

Bit 0 **SADD0**: Target address bit 0 (controller mode)
In 7-bit addressing mode (ADD10 = 0):
This bit is don't care
In 10-bit addressing mode (ADD10 = 1):
This bit should be written with bit 0 of the target address to be sent
Note: Changing these bits when the START bit is set is not allowed.

21.6.3 Own address 1register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:8]		OA1[7:1]							OA1[0]
rw					rw	rw		rw							rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own Address 1 enable
0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE** Own Address 1 10-bit mode
0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.
Note: This bit can be written only when OA1EN=0.

Bits 9:8 **OA1[9:8]**: Interface address
7-bit addressing mode: don't care
10-bit addressing mode: bits 9:8 of address
Note: These bits can be written only when OA1EN=0.

Bits 7:1 **OA1[7:1]**: Interface address
Bits 7:1 of address
Note: These bits can be written only when OA1EN=0.

Bit 0 **OA1[0]**: Interface address
7-bit addressing mode: don't care
10-bit addressing mode: bit 0 of address
Note: This bit can be written only when OA1EN=0.

21.6.4 Own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw			rw							

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own Address 2 enable
0: Own address 2 disabled. The received target address OA2 is NACKed.
1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN=0.
As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
bits 7:1 of address
Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.

21.6.5 Timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw								rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw								rw							

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings on page 460*) and for SCL high and low level counters (refer to *I2C controller initialization on page 474*).

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge in transmission mode.

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge SDA edge in transmission mode.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH+1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL+1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C is disabled (PE = 0).

21.6.6 Timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB [11:0]											
rw				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA [11:0]											
rw			rw	rw											

Bit 31 **TEXTEN**: Extended clock timeout enable
0: Extended clock timeout detection is disabled
1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B
This field is used to configure the cumulative clock extension timeout:
In controller mode, the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected
In target mode, the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected
$t_{LOW:EXT}$= (TIMEOUTB+1) x 2048 x t_{I2CCLK}
Note: These bits can be written only when TEXTEN=0.

Bit 15 **TIMOUTEN**: Clock timeout enable
0: SCL timeout detection is disabled
1: SCL timeout detection is enabled: when SCL is low for more than $t_{TIMEOUT}$ (TIDLE=0) or high for more than t_{IDLE} (TIDLE=1), a timeout error is detected (TIMEOUT=1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection
0: TIMEOUTA is used to detect SCL low timeout
1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)
Note: This bit can be written only when TIMOUTEN=0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A
This field is used to configure:
– The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE=0
$t_{TIMEOUT}$= (TIMEOUTA+1) x 2048 x t_{I2CCLK}
– The bus idle condition (both SCL and SDA high) when TIDLE=1
t_{IDLE}= (TIMEOUTA+1) x 4 x t_{I2CCLK}
Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 21.3: I2C implementation.

21.6.7 Interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r							r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (Target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Target mode)

This flag is updated when an address match event occurs (ADDR=1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a Stop condition is detected, or when PE=0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 10 **OVR**: Overrun/Underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced Start or Stop condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting *BERRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 7 **TCR**: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE=0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer Complete (controller mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE=0.

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a Stop condition is detected on the bus and the peripheral is involved in this transfer:

– either as a controller, provided that the STOP condition is generated by the peripheral.

– or as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 4 **NACKF**: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE=0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to ‘1’ by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).

Note: This bit is cleared by hardware when PE=0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to ‘1’ by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE=0.

21.6.8 Interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERT CF	TIM OUTCF	PECCF	OVRCF	ARLO CF	BERR CF	Res.	Res.	STOP CF	NACK CF	ADDR CF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.
Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.
Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 11 **PECCF**: PEC Error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.
Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 21.3: I2C implementation.

Bit 10 **OVRCF**: Overrun/Underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration Lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: Stop detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear
Writing 1 to this bit clears the ACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

21.6.9 PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 21.3: I2C implementation.

21.6.10 Receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]** 8-bit receive data
Data byte received from the I^2C bus.

21.6.11 Transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data
Data byte to be transmitted to the I^2C bus.
Note: These bits can be written only when TXE=1.

21.6.12 I2C register map

The table below provides the I2C register map and reset values.

Table 100. I2C register map and reset values

Offset	Register	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
0x0	**I2C_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	Res.	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	**I2C_CR2**	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						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
0x8	**I2C_OAR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0xC	**I2C_OAR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]	OA2[7:1]									Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	**I2C_TIMINGR**	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	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
0x14	**I2C_ TIMEOUTR**	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.		TIDLE	TIMEOUTA[11:0]												
	Reset value	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
0x18	**I2C_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	**I2C_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	**I2C_PECR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	**I2C_RXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 100. I2C register map and reset values (continued)

Offset	Register	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
0x28	**I2C_TXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

22 Universal synchronous asynchronous receiver transmitter (USART)

22.1 USART introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.

It supports synchronous one-way communication and half-duplex single wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). It also supports multiprocessor communications.

High speed data communication is possible by using the DMA (direct memory access) for multibuffer configuration.

22.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to give flexibility between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data, that can be enabled/disabled by software. FIFOs come with status flags for FIFOs states.
- A common programmable transmit and receive baud rate of up to 2Mbit/s with the clock frequency at 16 MHz and oversampling is by 8
- Dual clock domain with a dedicated kernel clock allowing baud rate programming independent from the PCLK reprogramming.
- Auto baud rate detection
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications
- Wakeup from mute mode (by idle line detection or address mark detection)

22.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T=0 and T=1 asynchronous protocols for Smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for Smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

22.4 USART implementation

Table 101 describes the USART implementation. It includes the LPUART for comparison.

Table 101. Instance implementation

Instance	Feature set
USART1	Full
LPUART1	Low power

Table describes the USART and LPUART implementation on STM32WB09xE device.

Table 102. USART / LPUART features

USART modes/features(1)	USART	LPUART
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (Master/Slave)	X	-
Smartcard mode	X	-
Single-wire half-duplex communication	X	X
IrDA SIR ENDEC block	X	-
LIN mode	X	-
Dual clock domain	X	X
Receiver timeout interrupt	X	-
Modbus communication	X	-
Auto baud rate detection	X	-
Driver Enable	X	X
USART data length	7, 8 and 9 bits	
Tx/Rx FIFO	X	X
Tx/Rx FIFO size	8	

1. X = supported.

22.5 USART functional description

Any USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX**: Receive Data Input.

  This is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

- **TX:** Transmit Data Output.

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and Smartcard modes, this I/O is used to transmit and receive the data.

Serial data are transmitted and received through these pins in normal USART mode. The frames are comprised of:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (7, 8 or 9 bits) least significant bit first
- 0.5, 1, 1.5, 2 Stop bits indicating that the frame is complete
- The USART interface uses a baud rate generator
- A status register (USART_ISR)
- Receive and transmit data registers (USART_RDR, USART_TDR). When FIFO mode is enabled, writing into USART_TDR adds one data to the transmit FIFO; and reading from USART_RDR removes one data from the receive FIFO.
- A baud rate register (USART_BRR)
- A guardtime register (USART_GTPR) in case of Smartcard mode.

Refer to *Section 22.7: USART registers* for the definitions of each bit.

The following pin is required to interface in synchronous mode and Smartcard mode:

- **SCLK:** This pin acts as Clock output in synchronous master and Smartcard modes. It acts as Clock input is Synchronous slave mode. In Synchronous Master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX. This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.

In Smartcard mode, SCLK output can provide the clock to the Smartcard.

- **NSS**: This pin acts as Slave Select input in Synronous slave mde.

The following pins are required in RS232 Hardware flow control mode:

- **nCTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS:** Request to send indicates that the USART is ready to receive data (when low).

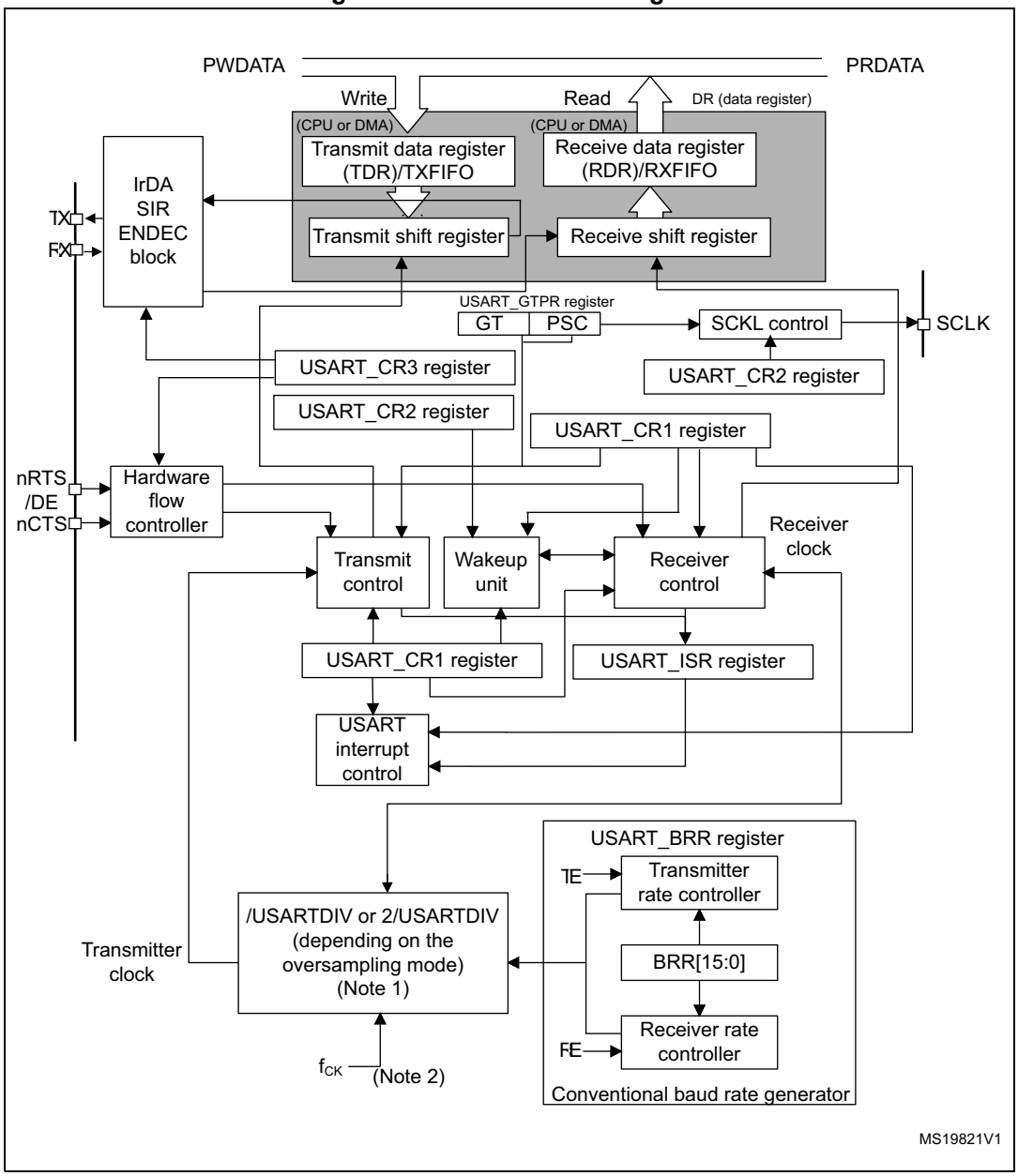
The following pin is required in RS485 Hardware control mode:

- **DE**: Driver Enable activates the transmission mode of the external transceiver.

Note: DE and nRTS share the same pin.

Note: NSS and nCTS share the same pin.

Figure 125. USART block diagram



1. For details on coding USARTDIV in the USARTx_BRR register, refer to *Section 22.5.5: Baud rate generation*.
2. f_{CK} is 16 MHz

22.5.1 USART character description

The word length can be selected as being either 7 or 8 or 9 bits by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 125*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and Autobaudrate (0x7F and 0x55 frames detection) are not supported.

In default configuration, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

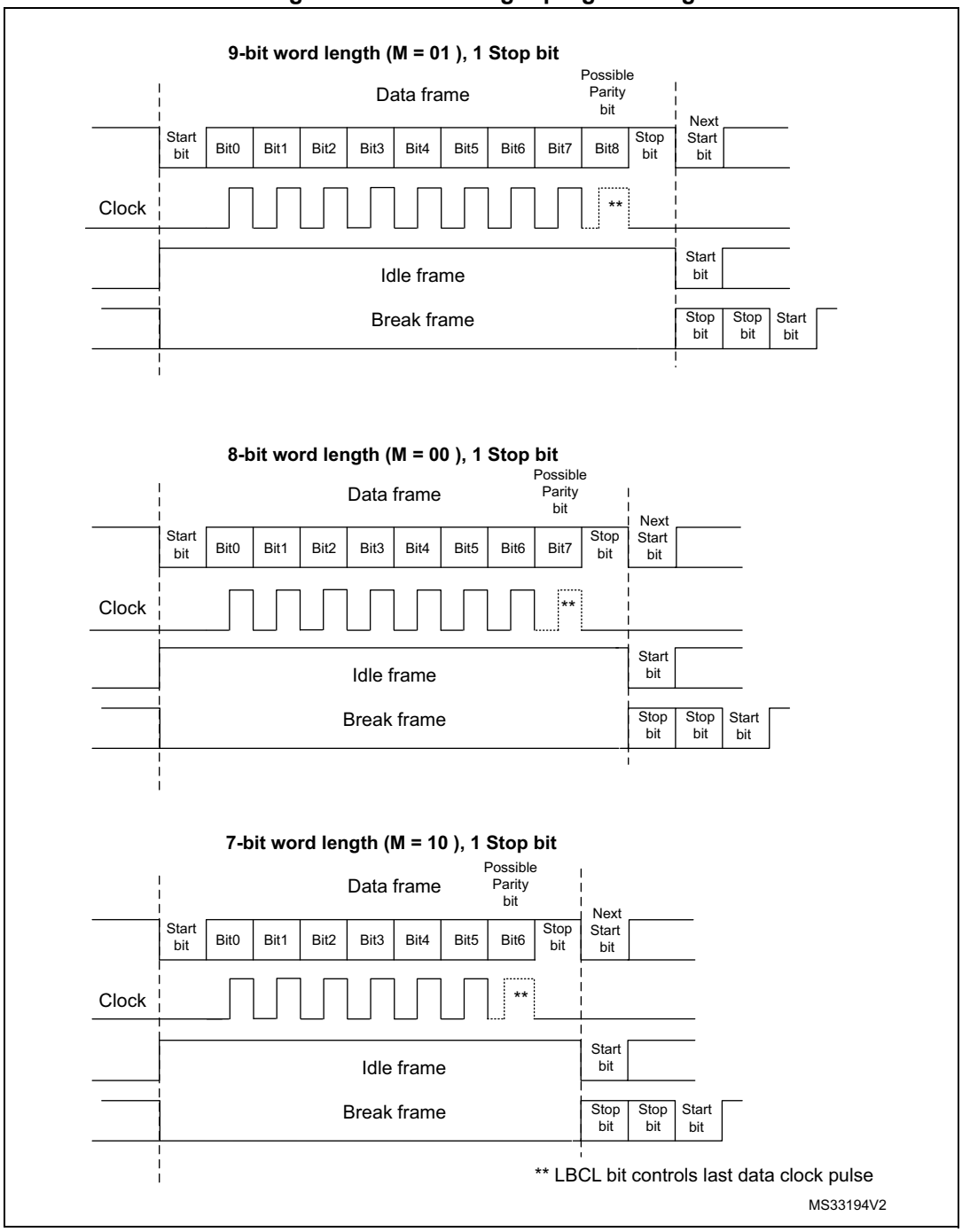
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 126. Word length programming



22.5.2 FIFOs and thresholds

The USART can operate in FIFO mode, with the FIFO buffers having a depth of 16 bytes.

The USART come with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO) . The FIFO mode is enabled by setting the bit 29 FIFOEN in USARTx_CR1 register.

The FIFO mode is supported only on UART, SPI and Smartcard modes.

Being 9 bits the maximum data word length, the TXFIFO is 9-bits wide. However the RXFIFO is by default 12-bits wide. This is due to the fact that the receiver does not only put the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO with its flags. But If you read RDR, you read just the data. The status flags are available in the USART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupt are triggered. These thresholds are programmed through bit fields RXFTCFG and TXFTCFG in USARTx_CR3 control register.

In this case:

- The receive interrupt in generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.
- The transmit interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

RXFIFO threshold

The RXFIFO threshold is configured using the RXFTCFG bits fields in the USARTx_CR3 register.

When the number of received data is equal to the programmed RXFTCFG, the flag RXFT in the USART_ISR register is set.

Having RXFT flag set means that there are RXFTCFG data received: 1 data in USARTx_RDR and (RXFTCFG - 1) data in the RXFIFO. So, when the RXFTCFG is programmed to «101» , the RXFT flag is set when 8 data are received: 7 data in the RXFIFO and 1 data in the USARTx_RDR. Consequently, the 9th received does not set the overrun flag.

22.5.3 Transmitter

The transmitter can send data words of either 7 or 8 or 9 bits depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the SCLK pin.

Character transmission

During an USART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the USARTx_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, data written to the transmit data register USART_TDR, is queued in the TXFIFO.

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters are frozen. The current data being transmitted is lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

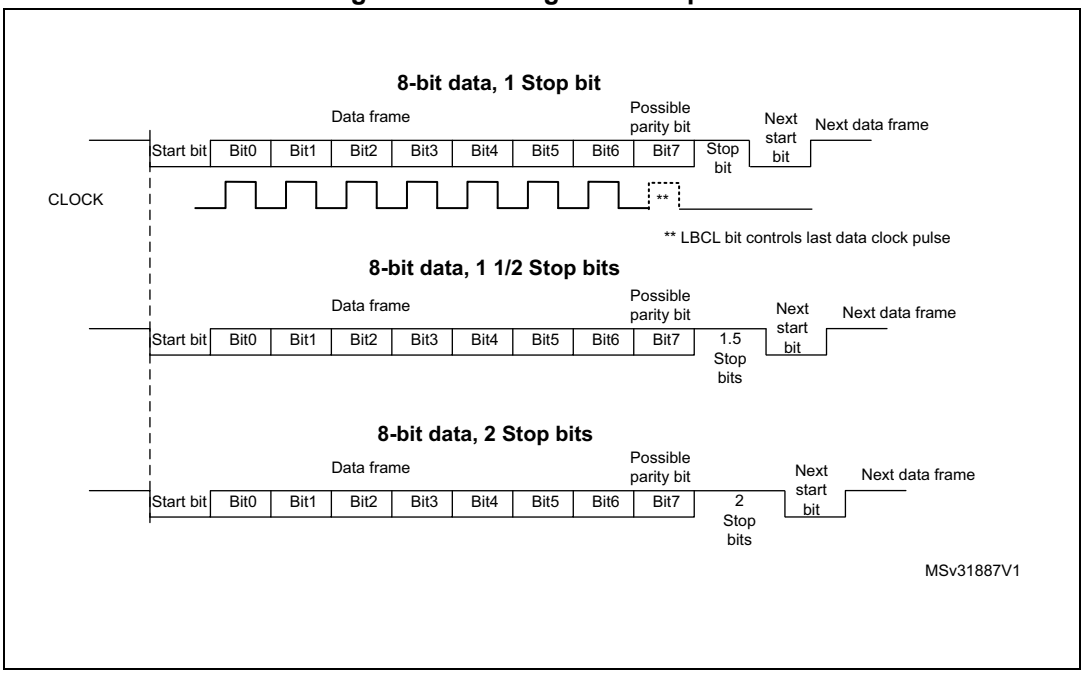
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in Smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Figure 127*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 127. Configurable stop bits



Character transmission procedure

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   a) When FIFO mode is disabled, writing a data in the USART_TDR clears the TXE flag.
   b) When FIFO mode is enabled, writing a data in the USART_TDR adds one data to the TXFIFO and write operations in the USART_TDR are made when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. After writing the last data into the USART_TDR register, wait until TC=1.
   a) When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   b) When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

Single byte communication

- When FIFO mode is disabled:

clearing the TXE flag is always performed by a write to the transmit data register.

The TXE flag is set by hardware and it indicates:

- The data has been moved from the USARTx_TDR register to the shift register and the data transmission has started.
- The USARTx_TDR register is empty.
- The next data can be written in the USARTx_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is taking place, a write instruction to the USARTx_TDR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the USARTx_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware and it indicates:
  - The TXFIFO is not full
  - The USART_TDR register is empty
  - The next data can be written in the USART_TDR register without overwriting the previous data. When a transmission is taking place, a write operation to the

USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO into the shift register at the end of the current transmission.

When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in USART_TDR . It is cleared when the TXFIFO is full.

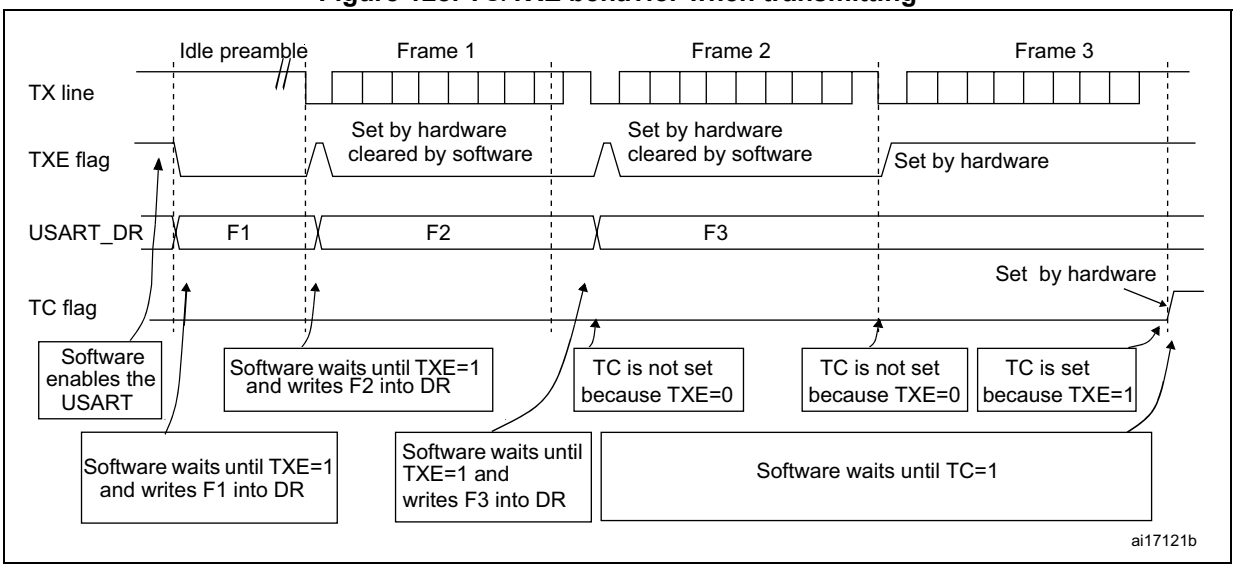
This flag generates an interrupt if the TXFNFIE bit is set.

Alternatively, interrupts can be generated and data can be written into FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed triger level.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data in the USART_TDR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low power mode (see *Figure 128: TC/TXE behavior when transmitting*).

Figure 128. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 129: Start bit detection when oversampling by 16 or 8*).

If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

22.5.4 Receiver

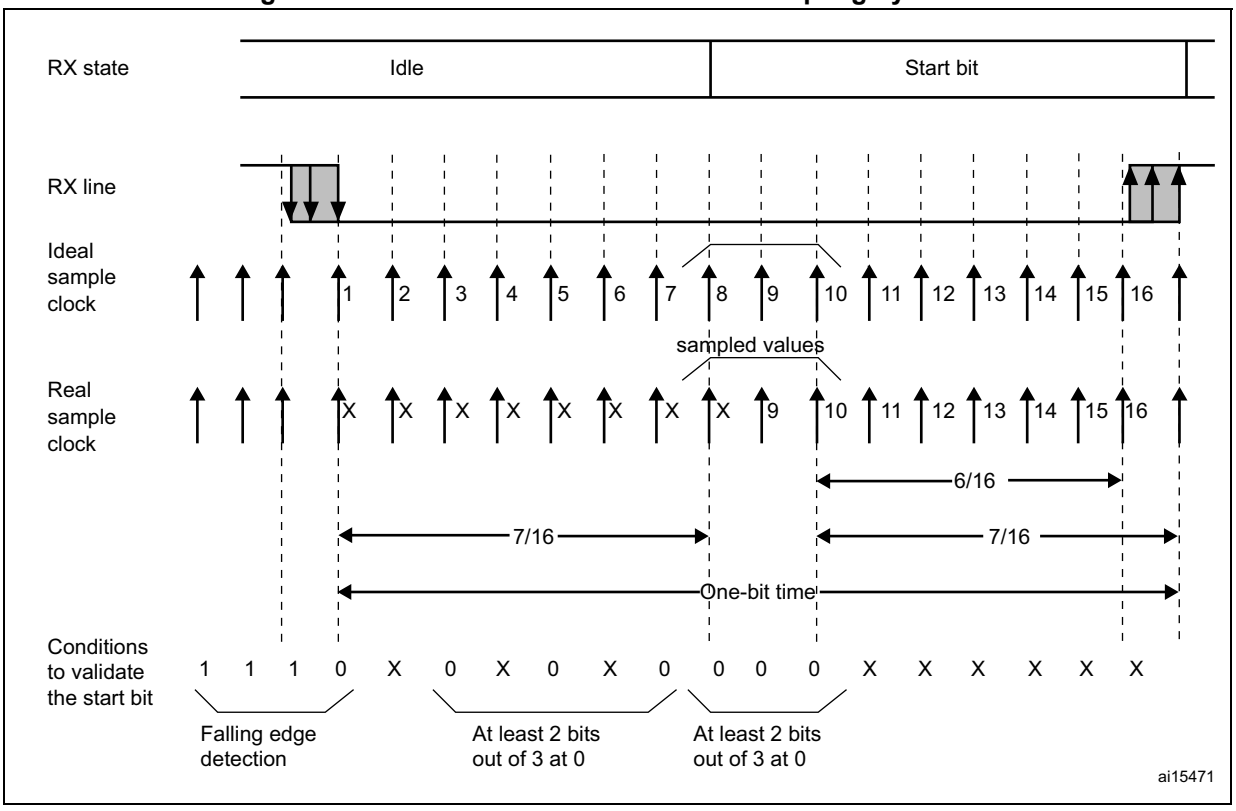
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 129. Start bit detection when oversampling by 16 or 8



Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set, interrupt generated if RXNEIE=1 (RXFNE flag set, interrupt generated if RXFNEIE=1 if FIFO mode is enabled) i)f the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NF noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither conditions a. or b. are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data shifts in least significant bit first (default configuration) through the RX pin.

Character reception procedure

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set indicating that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. A read of the USART_RDR getsthe oldest entry in the RXFIFO. When a data in received, it is stored in the RXFIFO, with error bits associated with that data.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error has been detected during reception.
- In multibuffer communication:
  - When FIFO mode is disabled, the RXNE is set after every byte received and is cleared by the DMA read of the Receive data Register.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is data in the RXFIFO to be read.
- In single buffer mode,
  - When FIFO mode is disabled: clearing the RXNE flag is performed by a software read to the USARTx_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USARTx_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read of the USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register. When the

RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as for a received data character plus an interrupt if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled:
  An overrun error occurs when a character is received when RXNE has not been reset. data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - The ORE bit is set.
  - The RDR content is not lost. The previous data is available when a read to
  - USARTx_RDR is performed.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXNEIE bit is set or EIE bit is set.
- FIFO mode enabled:
  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full. Data can not be transferred from the shift register to the USARTx_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty. An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - The ORE bit is set.
  - The first entry in the RXFIFO is not lost. It is available when a read to USART_RDR is performed.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE bit is set or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USARTx_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

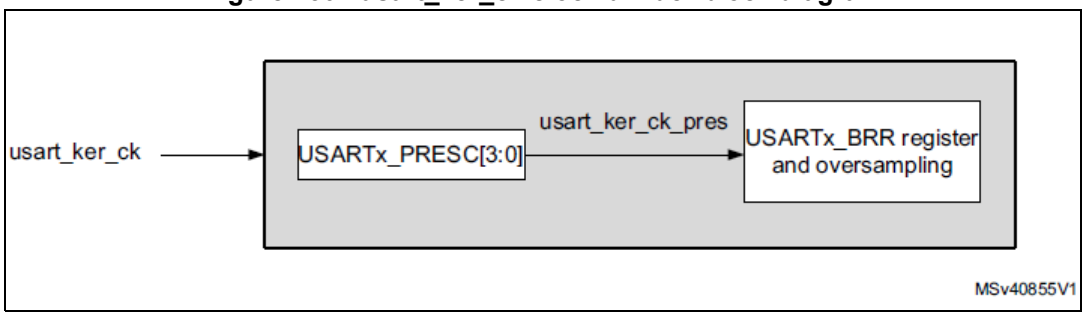
- *if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,*
- *if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source and the proper oversampling method

The clock source frequency is f_{CK}(16 MHz).

The f_{CK} can be divided by a programmable factor in the USARTx_PRESC register.

Figure 130. usart_ker_ck clock divider block diagram



The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This allows a trade of between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register and can be either 16 or 8 times the baud rate clock (*Figure 131* and *Figure 132*)

Depending on the application:

- Select oversampling by 8 (OVER8=1) to achieve higher speed (up to $f_{CKPRES}/8$). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 22.5.6: Tolerance of the USART receiver to clock deviation*)
- Select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum $f_{CKPRES}/16$.

where f_{CKPRES} is the USART input clock divided by a prescaler.

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. There are two options:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NF bit is set
- A single sample in the center of the received bit

  Depending on the application:
  - select the three samples' majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 103*) because this indicates that a glitch occurred during the sampling.
  - select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver's tolerance to clock deviations (see *Section 22.5.6: Tolerance of the USART receiver to clock deviation*). In this case the NF bit are never set.

When noise is detected in a frame:

- The NF bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NF bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI mode.

Note: Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0 ' by hardware.

Figure 131. Data sampling when oversampling by 16

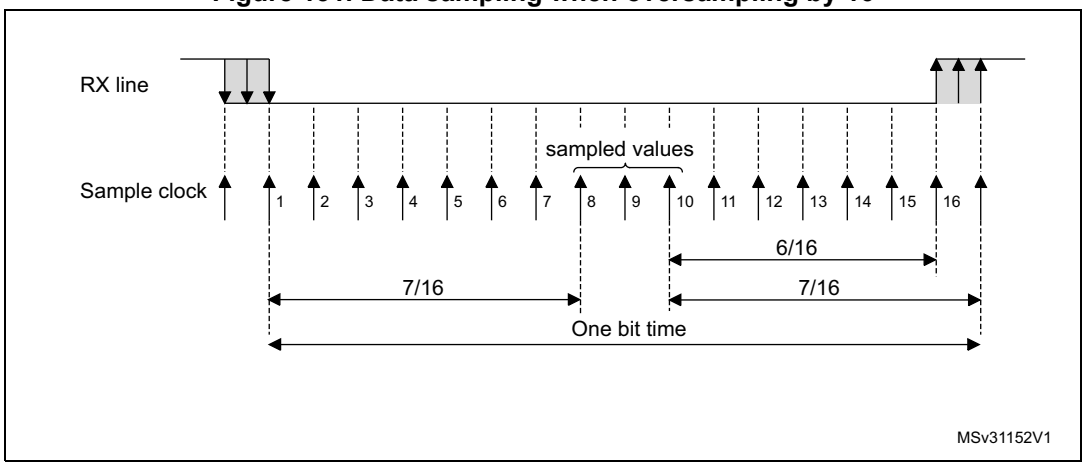


Figure 132. Data sampling when oversampling by 8

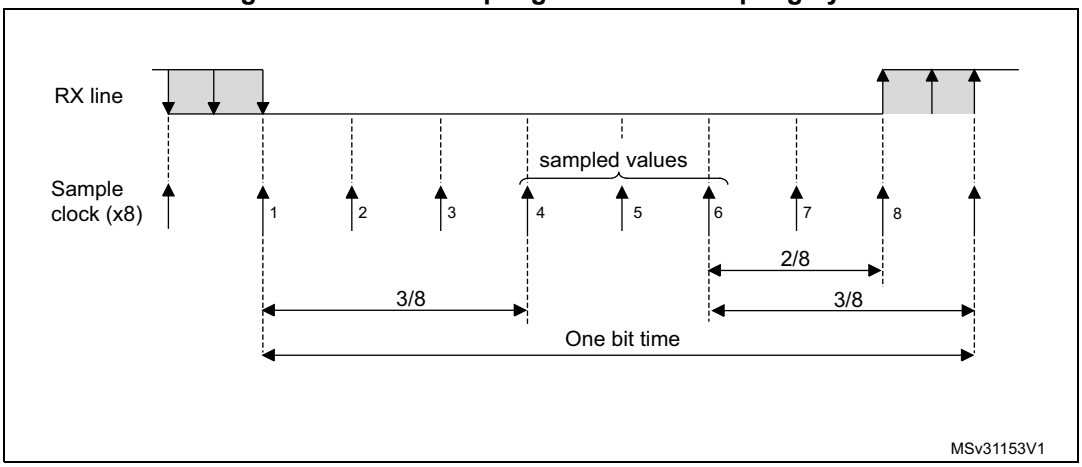


Table 103. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode)**: No sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit iis selected.
- ***1 stop bit***: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
- ***1.5 stop bits (Smartcard mode)***: When transmitting in Smartcard mode, the device must check that the data is correctly sent. Thus the receiver block must be enabled (RE =1 in the USART_CR1 register) and the stop bit is checked to test if the Smartcard has detected a parity error. In the event of a parity error, the Smartcard forces the data signal low during the sampling - NACK signal-, which is flagged as a framing error. Then, the FE flag is set with the RXNE (RXFNE in case FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be decomposed into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through. Refer to *Section 22.5.14: Receiver timeout* for more details.
- ***2 stop bits***: Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit the framing error flag is set. The second stop bit is not checked for framing error. The RXNE (RXFNE in case FIFO mode is enabled) flag is set at the end of the first stop bit.

22.5.5 Baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the USART_BRR register.

Equation 1: Baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the equation is:

$$\text{Tx/Rx baud} = \frac{f_{CKPRES}}{USARTDIV}$$

In case of oversampling by 8, the equation is:

$$\text{Tx/Rx baud} = \frac{2 \times f_{CKPRES}}{USARTDIV}$$

Equation 2: Baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

$$\text{Tx/Rx baud} = \frac{f_{CKPRES}}{USARTDIV}$$

fckpres is the USART clock which is the USART input clock divided by a prescaler onfigured in the USART_PRES regsiter.

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16d.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 baud with f_{CKPRES} = 8 MHz.

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR = USARTDIV = 833d = 0341h
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (1667d = 683h)
  BRR[3:0] = 3h >>1 = 1h
  BRR = 0x681

22.5.6 Tolerance of the USART receiver to clock deviation

The asynchronous receiver of the USART works correctly only if the total clock system deviation is less than the tolerance of the USART receiver. The causes which contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver's local oscillator
- DTCL: Deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL < USART\ receiver's\ tolerance$$

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 109: Error calculation for programmed baudrates at fck = 32,768 KHz*,

Table 110: Error calculation for programmed baudrates at fck = 16 MHz, depending on the following choices:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 104. Tolerance of the USART receiver when BRR [3:0] = 0000 (high-density devices)

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16	4.86	2.77	4.16

Table 105. Tolerance of the USART receiver when BRR[3:0] is different from 0000 (high-density devices)

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7	4.31	2.22	3.33

Note: The data specified in Table 104, Table 105 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M=01 or 9- bit times when M = 10).

22.5.7 Auto baud rate detection

The USART is able to detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance
- The system is using a relatively low accuracy clock source and this mechanism allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed (When oversampling by 16, the baud rate is between f_{CK}/65535 and f_{CK}/16. When oversampling by 8, the baudrate is between f_{CK}/65535 and f_{CK}/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be chosen. There are four modes based on different character patterns.

The modes can be chosen through the ABRMOD[1:0] field in the USART_CR2 register. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are:

- **Mode 0**: Any character starting with a bit at 1.
  In this case the USART measures the duration of the Start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.
  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, ensuring better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).
  In this case, the baudrate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to Bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.
  In this case, the baudrate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit0 is sampled at BRs, Bit1 to Bit6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate transition of RX line. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled, in case of auto baud rate error, the ABRE flag is set with RXNE and FE.

When FIFO management is enabled, in case of auto baud rate error, the ABRE flag is set with RXFNE and FE.

In case of FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launch the auto baud rate detection, the user should make sure that the RXFIFO is empty using the RXFNE flag in USARTx_ISR register.

Note: The BRR value may be corrupted if the USART is disabled (UE=0) during an auto baud rate operation.

22.5.8 Multiprocessor communication

It is possible to perform multiprocessor communication with the USART (with several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output connected to the RX inputs of the other USARTs. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In order to use the mute mode feature, the MME bit must be set in the USARTx_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two UCLK cycles) otherwise mute mode might remain active.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

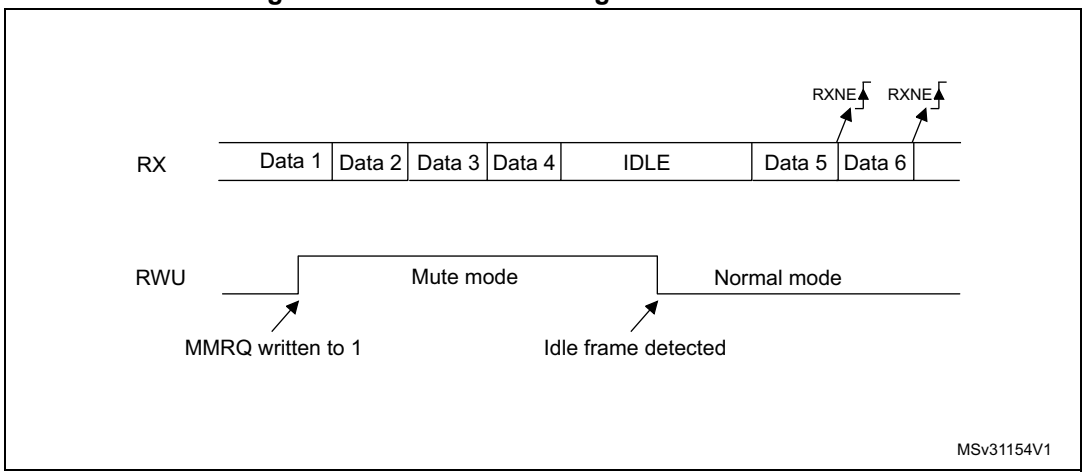
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 133*.

Figure 133. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering mute mode.

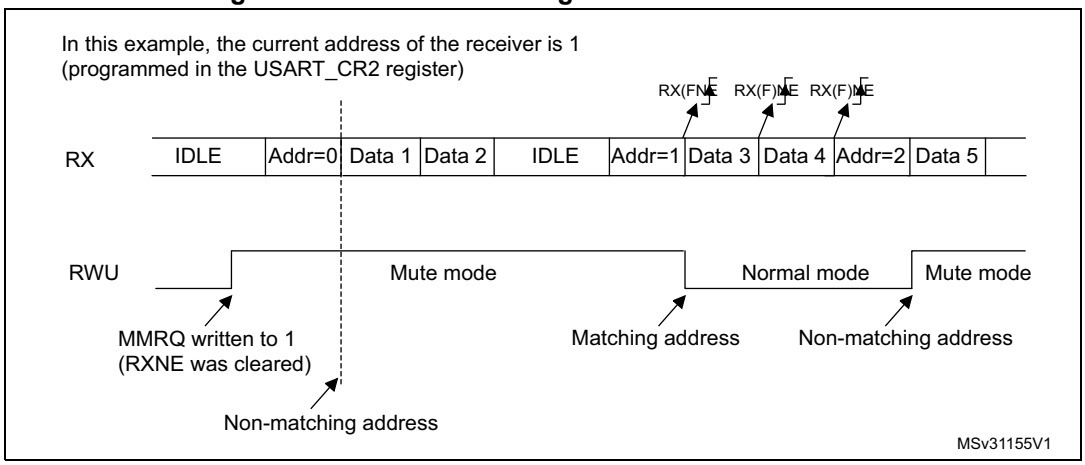
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 134*.

Figure 134. Mute mode using address mark detection



22.5.9 Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half duplex, block transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE=1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

22.5.10 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 106*.

Table 106. Frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

22.5.11 LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 22.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure explained in *Section 22.5.2* has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0 bits as a break character. Then 2 bits of value ‘1 are sent to allow the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as ‘0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 135: Break detection in LIN mode (11-bit break length - LBDL bit is set)*.

Examples of break frames are given in *Figure 136: Break detection in LIN mode vs. Framing error detection*.

Figure 135. Break detection in LIN mode (11-bit break length - LBDL bit is set)

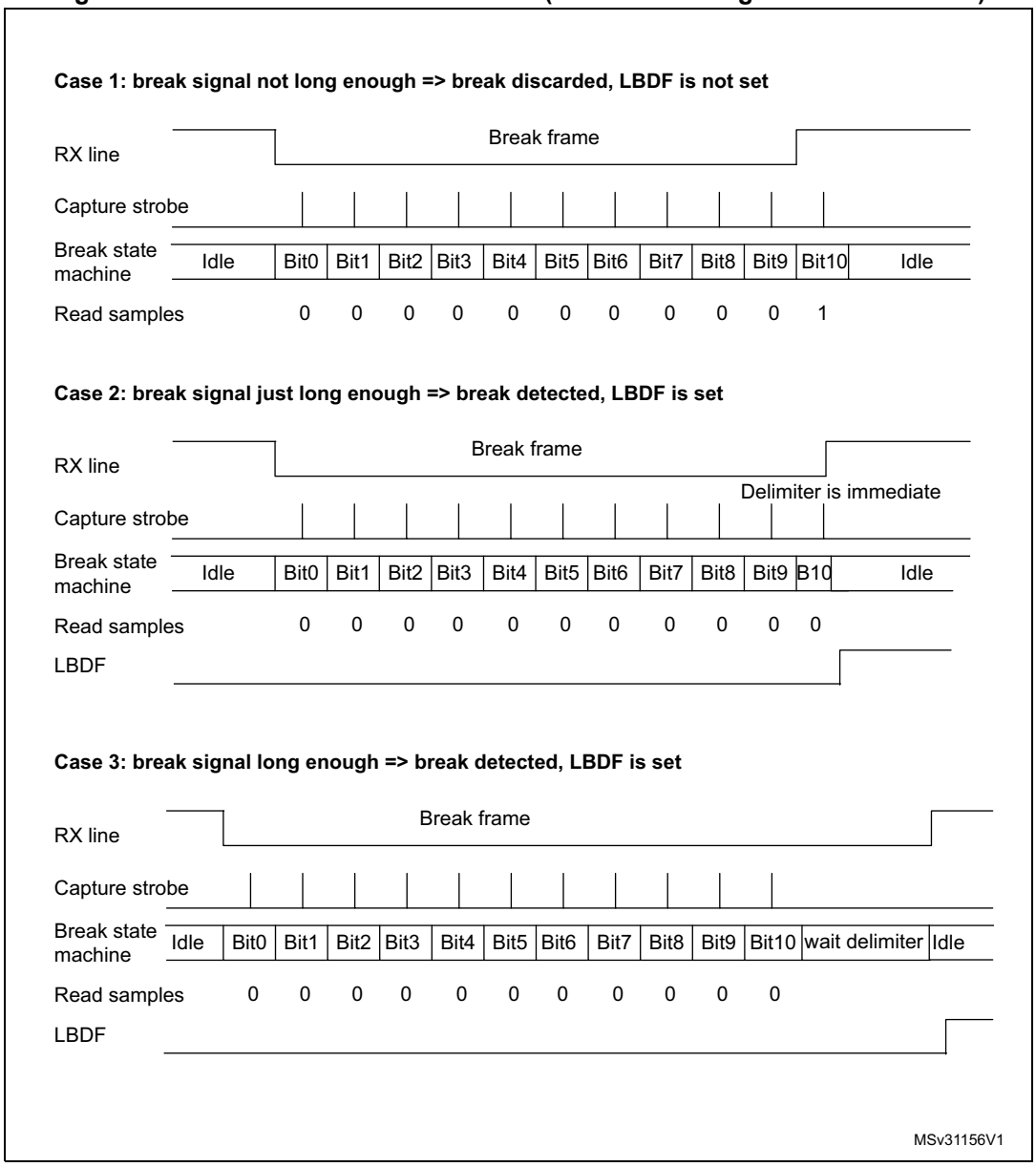
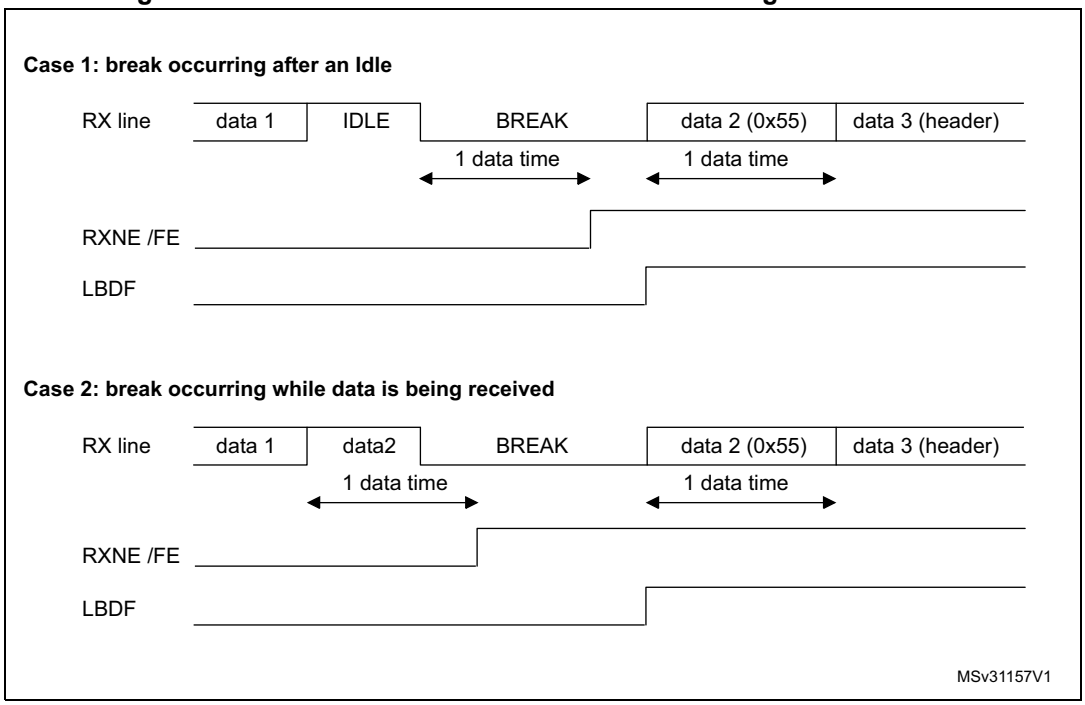


Figure 136. Break detection in LIN mode vs. Framing error detection



22.5.12 USART synchronous mode

Master Mode

The synchronous master mode is selected by writing the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 137*, *Figure 138* and *Figure 139*).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous master mode the USART transmitter works exactly like in asynchronous mode. But as SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the SCLK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data is being transmitted (the data

register USART_DR written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 137. USART example of synchronous Master transmission

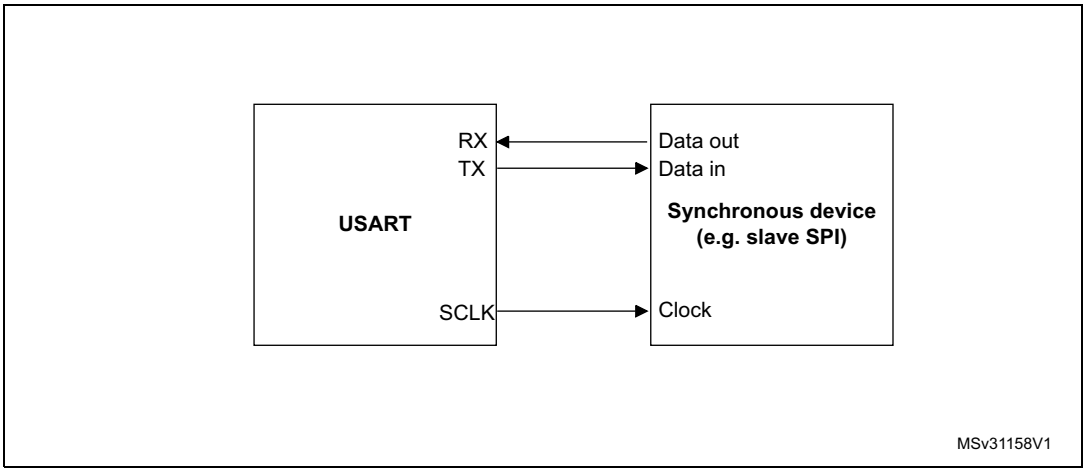


Figure 138. USART data clock timing diagram (M=0)

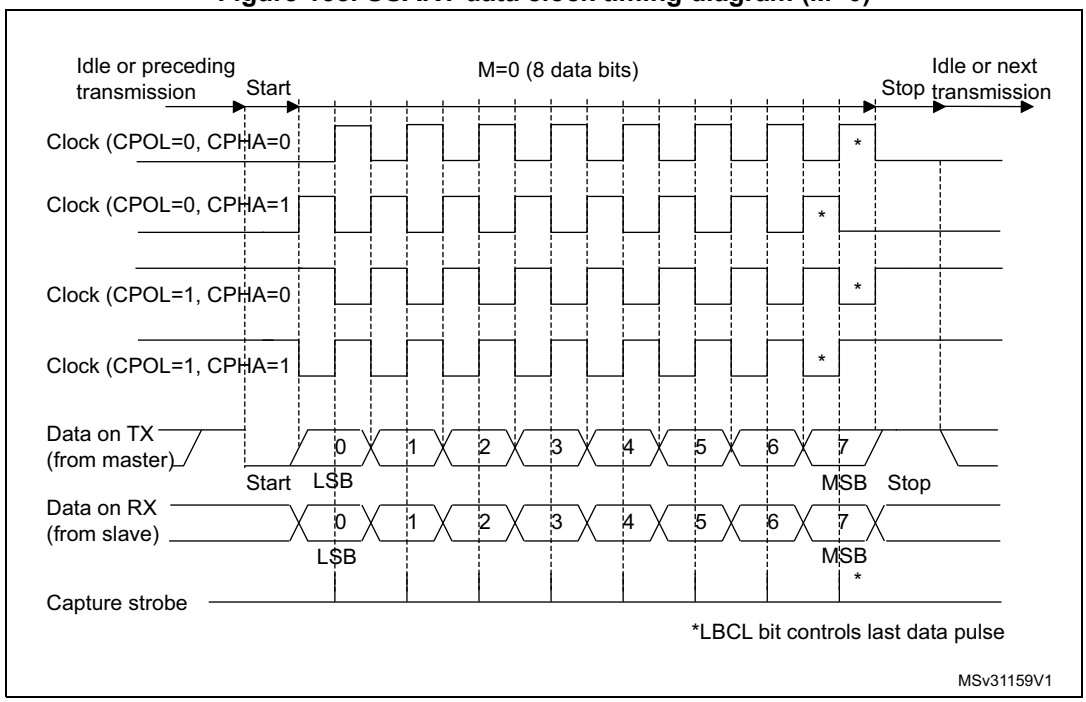


Figure 139. USART data clock timing diagram (M bits = 01)

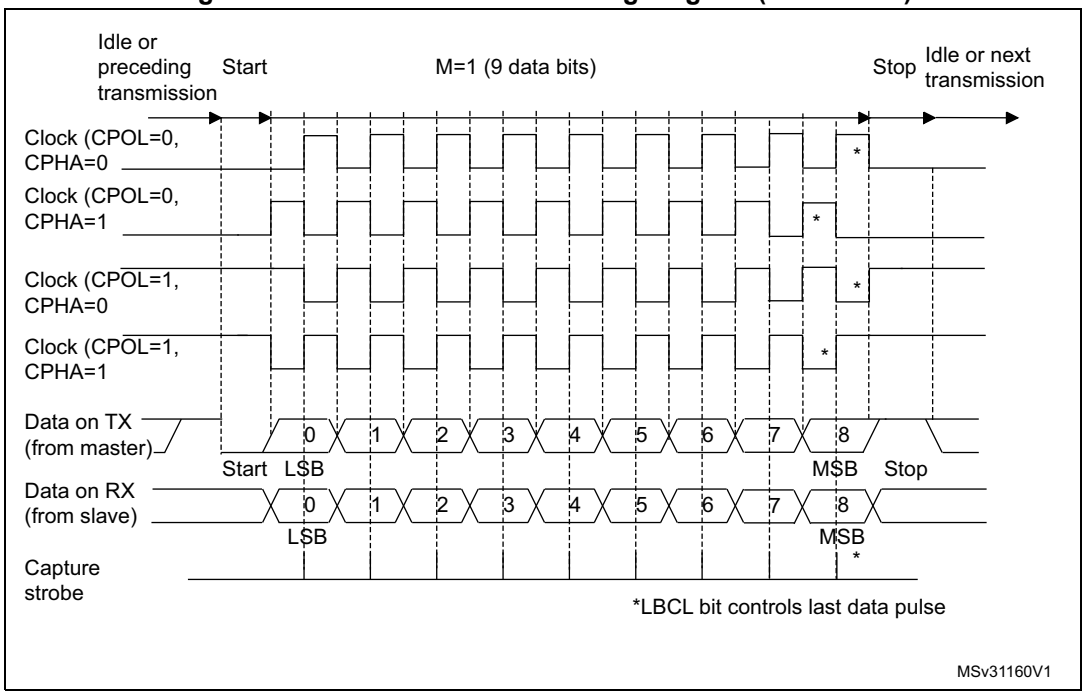
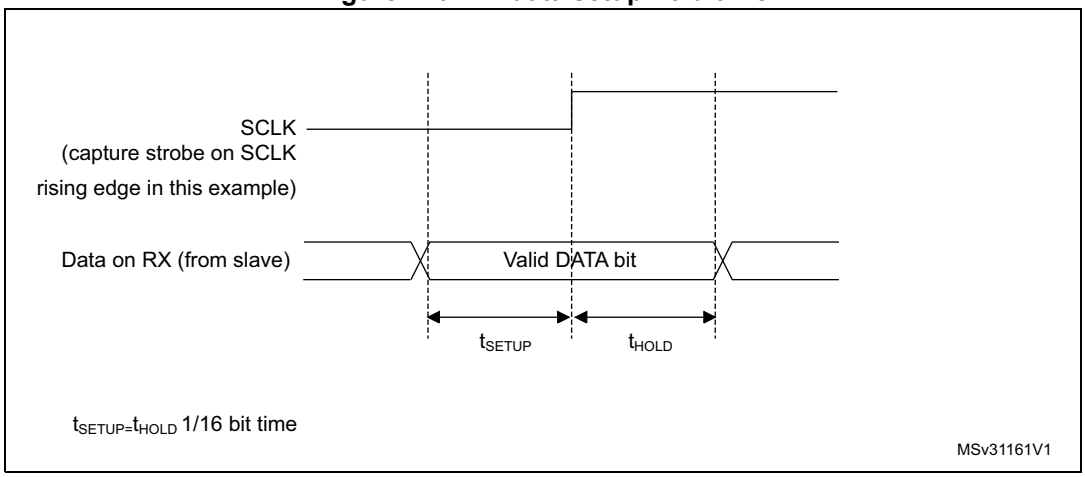


Figure 140. RX data setup/hold time



Slave mode

The synchronous slave mode is selected by writing the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The SCLK pin is the input of the USART in slave mode.

Note: When the peripheral is used in SPI slave mode, the peripheral clock source (fck_pres) must be greater than 3xSCLK input clock.

The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 137*, *Figure 138* and *Figure 139*).

In slave transmission mode, an underun error flag is available. This flag is set when the first clock for data transmission appears while the software has not yet loaded any value into USARTx_TDR.

The slave supports the hardware and software NSS management.

Slave Select (NSS) pin management:

Hardware or software slave select management can be set using the DIS_NSS bit in the

USART_CR2 register.

- Software NSS management (DIS_NSS = 1)

SPI slave is always selected and NSS input pin is ignored .

The external NSS pin remains free for other application uses.

- Hardware NSS management (DIS_NSS = 0)

The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly

Note: When in SPI slave mode, the USART must be enabled before the master starts communication (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI Slave underrun error

When an underrun error occurs, the SPI slave sends the last data until the underrrun error flag is cleared in software.

The underrun flag is set at the beginning of the frame.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In underrun condition, it is allowed to write the TDR register. Clearing the underrun error would allow sending the new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the data is written in the USARTx_TDR too close to the first SCLK transmision edge. To avoid this underrun error, the USART_TDR should be written 3 UCLK cycles before the first first SCLK edge.

22.5.13 Single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflicts on the line must be managed by software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: As the TX line and the RX lines are connected together, all the transmitted data are stored in the RX FIFO as the data received from an external device. The software has to take care to discard its "own" information after a transmit phase.

In half-duplex mode, it is always wise to read back the transmitted data to check if they are correct as there is no hardware protection against possible collision between nodes.

If the software does not want to have the RX FIFO storing the transmitted value then it has to disable the receiver part while transmitting (by clearing the RE bit in USART_CR1 register).

22.5.14 Receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bit fields in the USARTx_RTOR register.

The receiver timeout counter starts counting

- From the end of the stop bit in case STOP = 00 and STOP = 11
- From the end of the second stop bit in case STOP = 10.
- From the beginning of the stop bit in case STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USARTx_ISR register is set. and a timeout is generated if RTOIE bit in USARTx_CR1 register is set.

22.5.15 Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to *Section 22.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit may be set in order to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

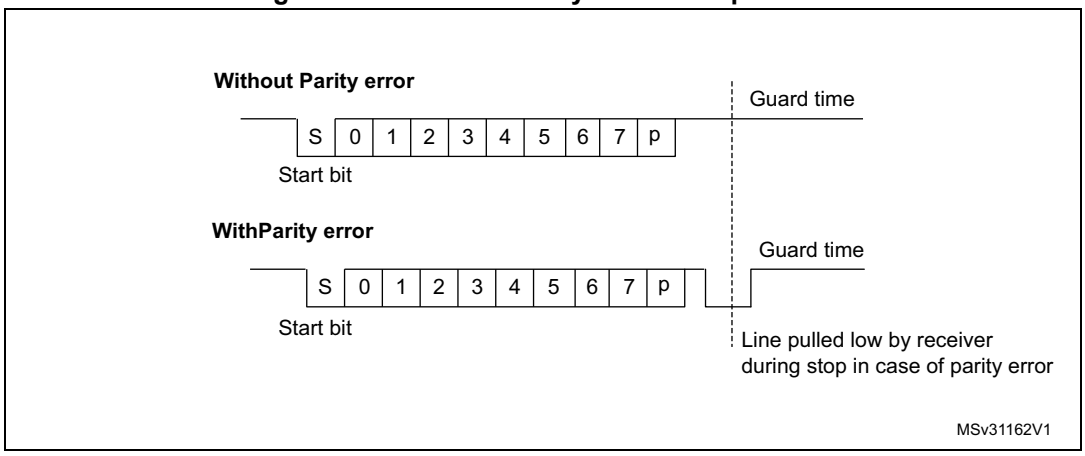
The USART should be configured as:

- 8 bits plus parity: where M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting abd receiving data: where STOP=11 in the USART_CR2 register. It is also possible to choose the 0.5 stop bit for receiving.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 141 shows examples of what can be seen on the data line with and without parity error.

Figure 141. ISO 7816-3 asynchronous protocol



When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol. The number of retries is programmed in the SCARCNT bit field. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bit field, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn’t repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the

desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).

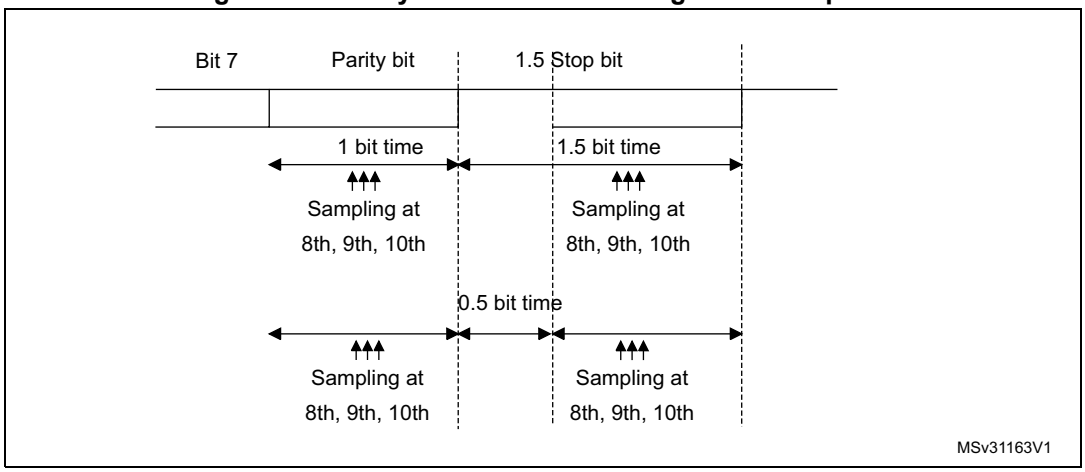
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: A break character is not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 142 details how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 142. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the SCLK output. In Smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART_. SCLK frequency can be programmed from $f_{CKPRES}/2$ to $f_{CKPRES}/62$, where f_{CKPRES} is the peripheral input clock divided by a programmed prescaler.

Block mode (T=1)

In T=1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the UART_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time) - 11 value, in order to allow the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baudtime units. If the Smartcard doesn't send a new character in less than the CWT period after the end of the previous character, the USART signals this to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the Smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value (0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

Note: When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character is ‘03’ and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card didn’t answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103 -> inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B -> direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

22.5.16 IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 22.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 143*).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.
- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 143*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods is accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bits in the USART_CR2 register must be configured to "1 stop bit".

IrDA low-power mode

Transmitter

In low-power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

Receiver:

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).

Figure 143. IrDA SIR ENDEC- block diagram

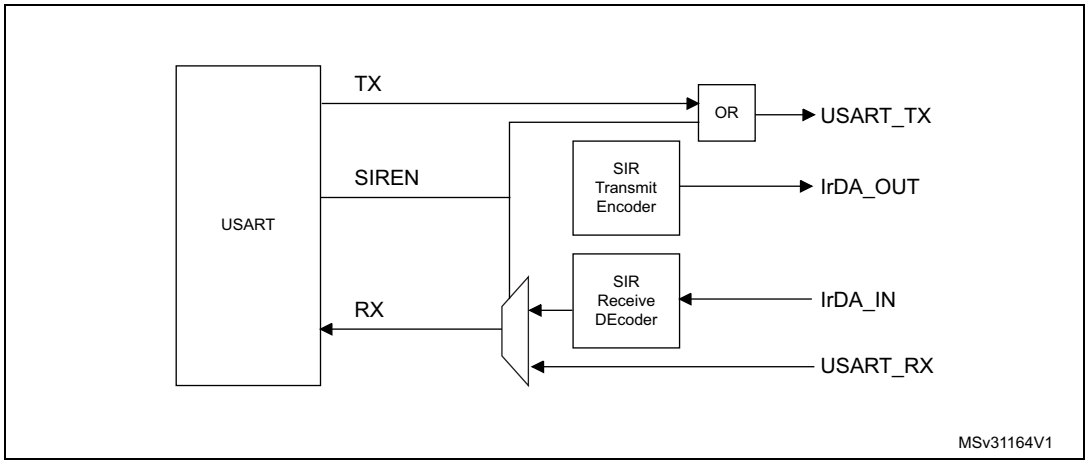
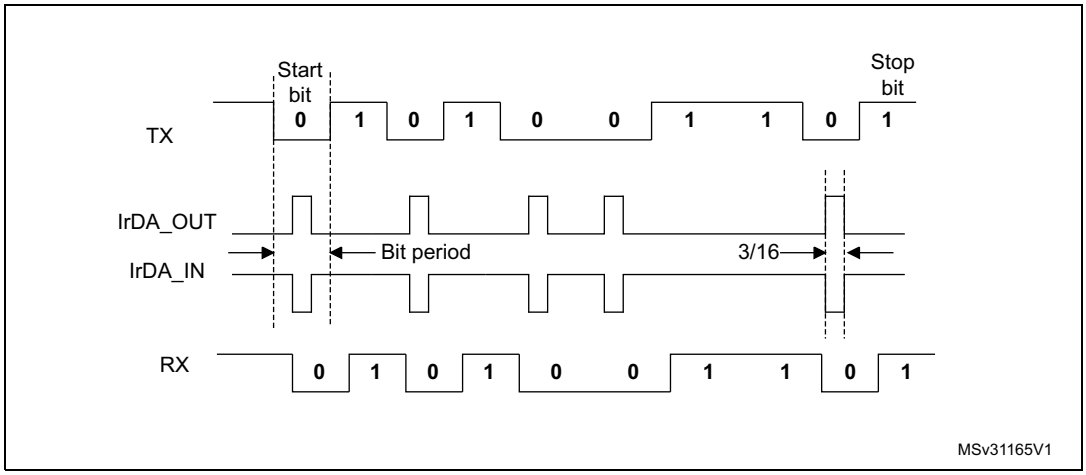


Figure 144. IrDA data modulation (3/16) -Normal Mode



22.5.17 Continuous communication using DMA

The USART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 22.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in or Section 22.5.4: Receiver. To perform continuous communication, When FIFO is disabled, you can clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

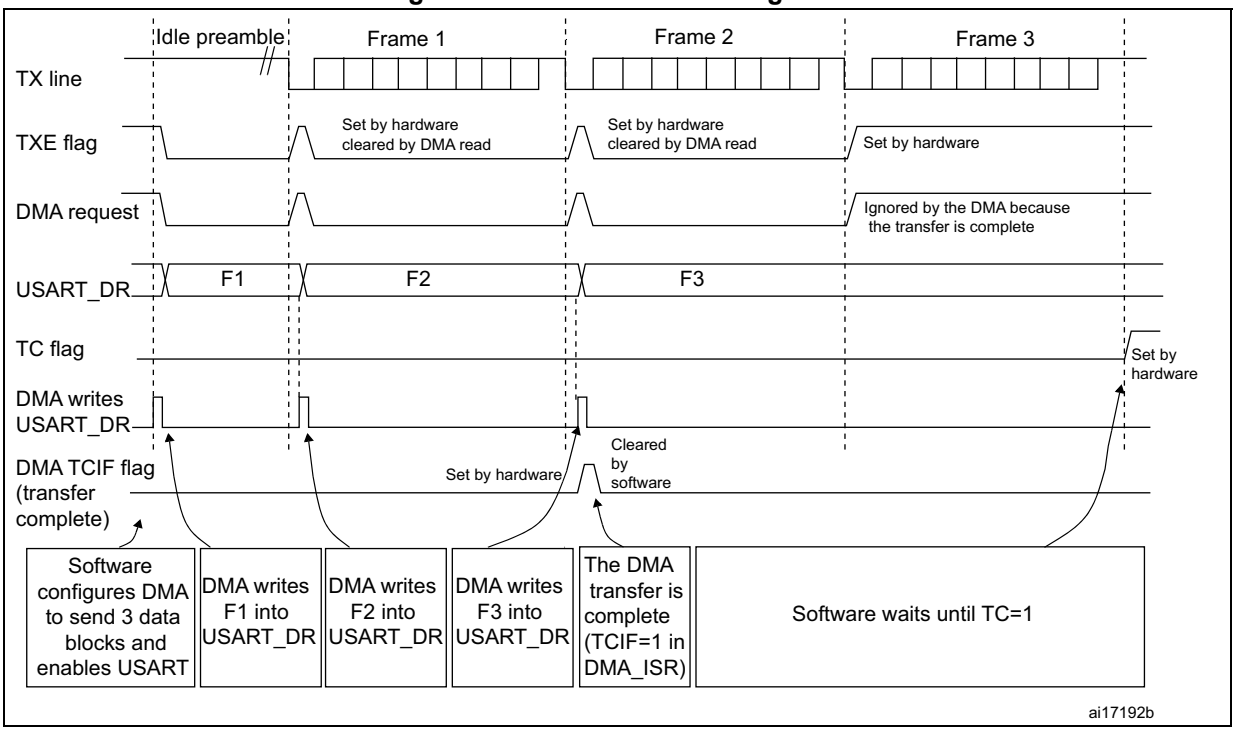
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to *Section 10: DMA controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering Deepstop mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 145. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

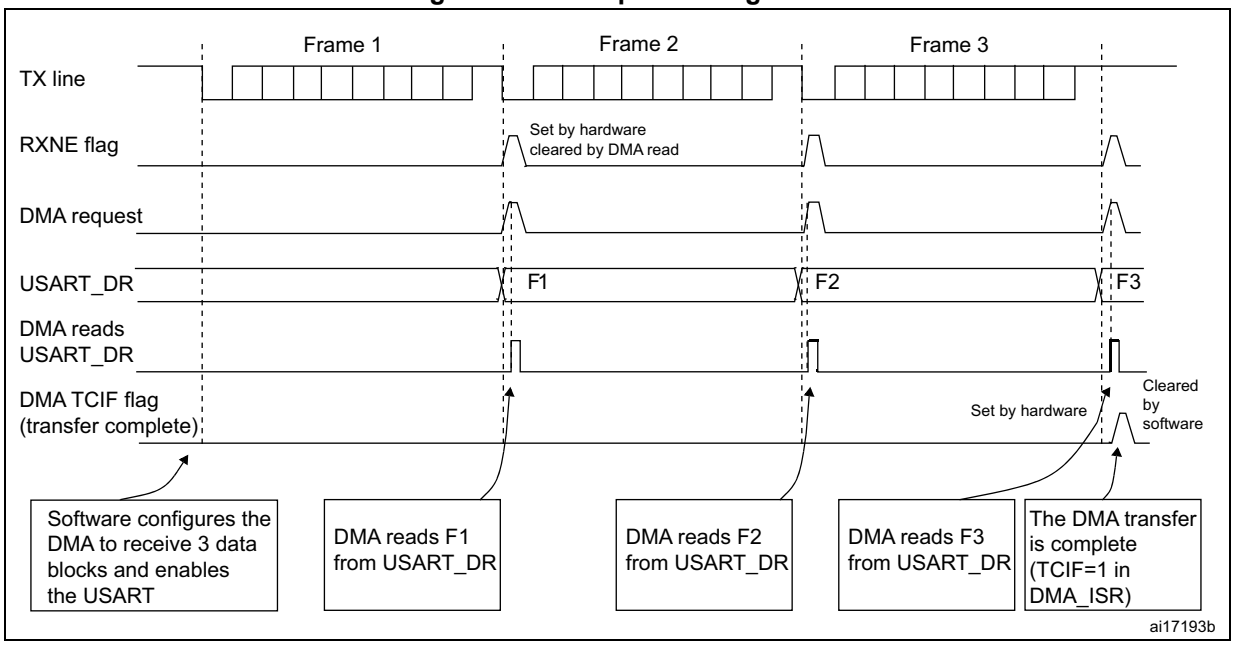
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data is loaded from the USART_RDR register to a SRAM area configured using the DMA peripheral (refer to *Section 10: DMA controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 146. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

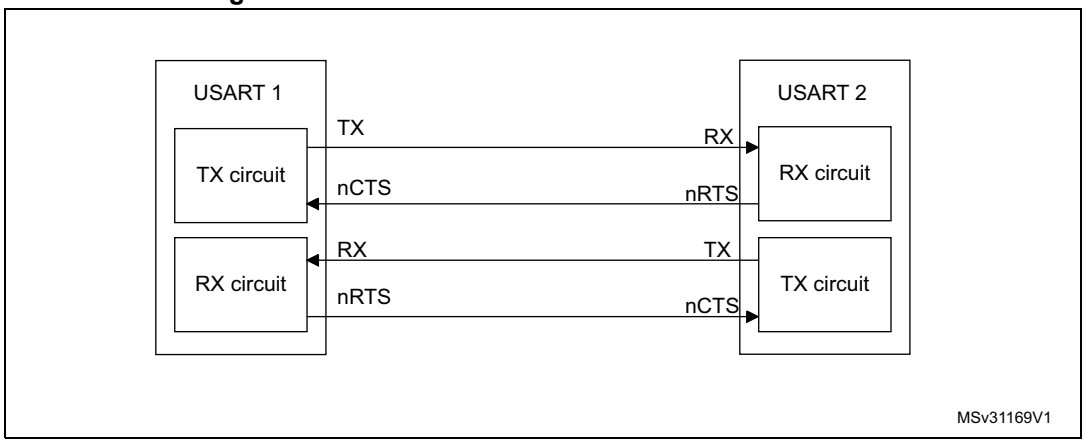
Error flagging and interrupt generation in multibuffer communication

In multibuffer communication if any error occurs during the transaction the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

22.5.18 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. *Figure 147* shows how to connect 2 devices in this mode:

Figure 147. Hardware flow control between 2 USARTs

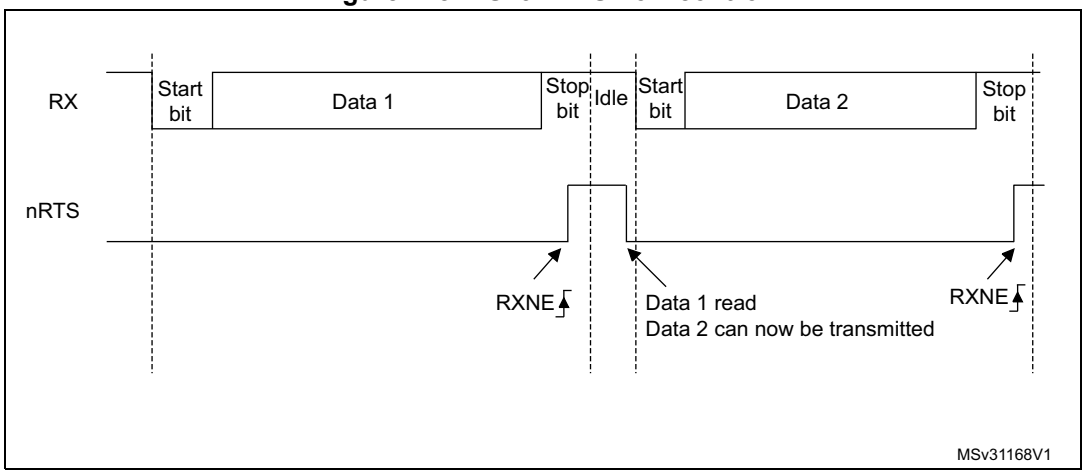


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the USART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 148* shows an example of communication with RTS flow control enabled.

Figure 148. RS232 RTS flow control



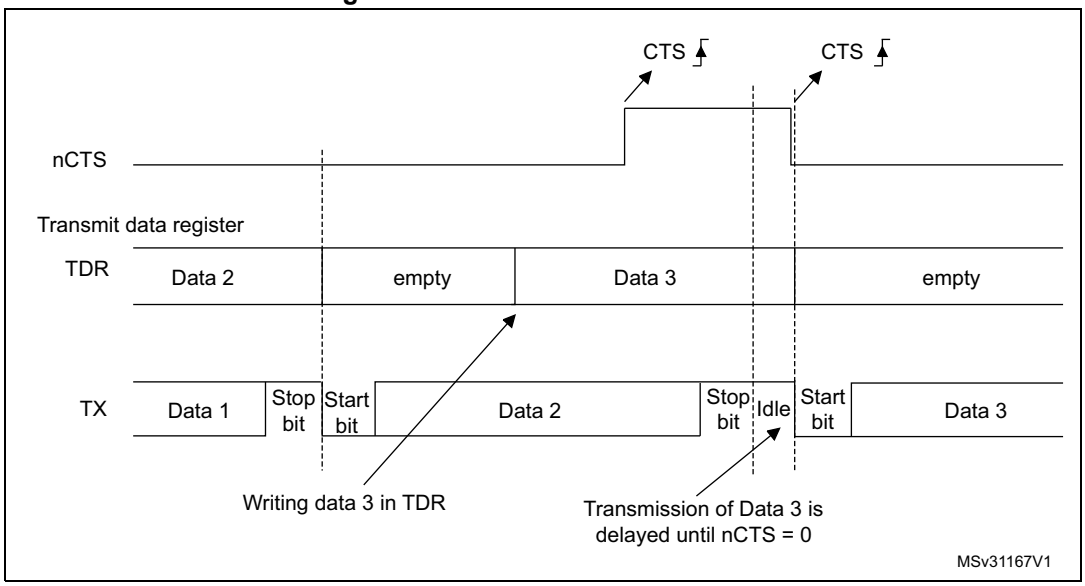
Note: When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 149* shows an example of communication with CTS flow control enabled.

Figure 149. RS232 CTS flow control



Note: For correct behavior, nCTS must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 Driver Enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the START bit. It is programmed using the DEAT [4:0] bit fields in the USART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bit fields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

22.6 USART interrupts

Table 107. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
Transmit FIFO Not Full	TXFNF	TXFNFIE
Transmit FIFO Empty	TXFE	TXFEIE
CTS interrupt	CTSIF	CTSIE
Transmission Complete	TC	TCIE
Transmission Complete Before Guard Time	TCBGT	TCBGTIE
Receive data register not empty (data ready to be read)	RXNE	RXNEIE
Receive FIFO Not Empty	RXFNE	RXFNEIE
Receive FIFO Full	RXFF	RXFFIE
Overrun error detected	ORE	RXNEIE/RXF-NEIE
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
LIN break	LBDF	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NF or ORE or FE	EIE
Character match	CMF	CMIE
Receiver timeout error	RTOF	RTOIE
End of Block	EOBF	EOBIE
SPI slave underrun error	UDR	EIE

These events generate an interrupt if the corresponding Enable Control Bit is set.

22.7 USART registers

Refer to *Section 1.5 on page 35* for a list of abbreviations used in register descriptions.

22.7.1 Control register 1 (USARTx_CR1)

Address offset: 0x00

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE/ TXFNFIE	TCIE	RXNEIE /RXFNEIE	IDLEIE	TE	RE	Res.	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **RXFFIE** :RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An USART interrupt is generated when RXFF=1 in the USART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 30 **TXFEIE** :TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An USART interrupt is generated when TXFE=1 in the USART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 29 **FIFOEN** :FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bit field can only be written when the USART is disabled (UE=0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit, with bit 12 (M0) determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 Start bit, 8 Data bits, n Stop bit
M[1:0] = 01: 1 Start bit, 9 Data bits, n Stop bit
M[1:0] = 10: 1 Start bit, 7 Data bits, n Stop bit
This bit can only be written when the USART is disabled (UE=0).

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and Autobaudrate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Section 22.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bit field can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Refer to Section 22.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Refer to Section 22.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the USART. When set, the USART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit, with bit 28 (M1) determine the word length. It is set or cleared by software. See Bit 28 (M1)description.

This bit can only be written when the USART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bit field can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bit field can only be written when the USART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bit field can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever PE=1 in the USART_ISR register

Bit 7 **TXEIE/TXFNFIE**: Transmit data regsiter empty/TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever TXE/TXFNF =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever TC=1 in the USART_ISR register

Bit 5 **RXNEIE/RXFNEIE**: Receive data register not empty/RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever IDLE=1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 Reserved, must be kept at reset value

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the USART is kept, but all the status flags, in the USART_ISR are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low power mode

1: USART enabled

Note: In order to go into low power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

Note: in Smartcard mode, (SCEN = 1), the SCLK is always available when CLKEN = 1, regardless of the UE bit value.

22.7.2 Control register 2 (USARTx_CR2)

Address offset: 0x04

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS.	Res.	Res.	SLVEN.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:28 **ADD[7:4]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.
This is used in multiprocessor communication during Mute mode, for wakeup with 7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. It may also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.
This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.
This is used in multiprocessor communication during Mute mode, for wakeup with address mark detection.
This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.
0: Receiver timeout feature disabled.
1: Receiver timeout feature enabled.
When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.
00: Measurement of the start bit is used to detect the baud rate.
01: Falling edge to falling edge measurement. (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)
10: 0x7F frame detection.
11: 0x55 frame detection
This bit field can only be written when ABREN = 0 or the USART is disabled (UE=0).

Note: If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bit field can only be written when the USART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bit field can only be written when the USART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: TX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the TX line.

This bit field can only be written when the USART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: RX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the RX line.

This bit field can only be written when the USART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART.

This bit field can only be written when the USART is disabled (UE=0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support LIN mode, this bit is reserved and forced by hardware to '0'.Refer to Section 22.4: USART implementation.

Bits 13:12 **STOP[1:0]**: STOP bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bit field can only be written when the USART is disabled (UE=0).

Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the SCLK pin.

0: SCLK pin disabled

1: SCLK pin enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Note: In Smartcard mode, in order to provide correctly the SCLK clock to the smartcard, the steps below must be respected:

- UE = 0
- SCEN = 1
- GTPR configuration
- CLKEN= 1
- UE = 1

Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on SCLK pin outside transmission window

1: Steady high value on SCLK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 138* and *Figure 139*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the SCLK pin

1: The clock pulse of the last data bit is output to the SCLK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable
Break interrupt mask (break detection using break delimiter).
0: Interrupt is inhibited
1: An interrupt is generated whenever LBDF=1 in the USART_ISR register
Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length
This bit is for selection between 11 bit or 10 bit break detection.
0: 10-bit break detection
1: 11-bit break detection
This bit can only be written when the USART is disabled (UE=0).
Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the USART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**
When the DSI_NSS bit is set, the NSS pin input is ignored.
0: SPI slave selection depends on NSS input pin.
1: SPI slave is always selected and NSS input pin is ignored.
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous Slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value

Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

22.7.3 Control register 3 (USARTx_CR3)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG			RXFTIE.	RXFTCFG			TCBGTIE	TXFTIE	Res.	Res.	Res.	SCARCNT2:0]			Res.
rw			rw	rw			rw	rw				rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
010:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Remaining combinations: Reserved.

Bit28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An USART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
010:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Remaining combinations: Reserved.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An USART interrupt is generated whenever TCBGT=1 in the USARTx_ISR register

Note: If the USART does not support the Smartcard mode, this bit is reserved and forced by hardware to '0'.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An USART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bits 22:20 Reserved, must be kept at reset value.

Bit 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bit-field specifies the number of retries in transmit and receive, in Smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bit field must be programmed only when the USART is disabled (UE=0).
When the USART is enabled (UE=1), this bit field may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmit mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Refer to Section 22.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit allows the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Section 22.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for Smartcard mode)
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.
This bit can only be written when the USART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 : **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data is written directly in USARTx_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.
This bit can only be written when the USART is disabled (UE=0).

Note: This control bit allows checking the communication flow w/o reading the data

Bit 11 **ONEBIT**: One sample bit method enable

This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (NF) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE=0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the USART is disabled (UE=0)

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE=0).

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NF=1or UDR = 1 in the USART_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE=1 or ORE=1 or NF=1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

22.7.4 Baud rate register (USARTx_BRR)

This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **BRR[15:4]**

BRR[15:4] = USARTDIV[15:4]

Bits 3:0 **BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].
When OVER8 = 1:
BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
BRR[3] must be kept cleared.

22.7.5 Guard time and prescaler register (USARTx_GTPR)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw								rw							

Bits 31:16 Reserved, must be kept at reset value

Bits 15:8 **GT[7:0]**: Guard time value

This bit-field is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bit field can only be written when the USART is disabled (UE=0).

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

In IrDA Low-power and normal IrDA mode:

PSC[7:0] = IrDA Normal and Low-Power Baud Rate

Used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

In Smartcard mode:

PSC[4:0]: Prescaler value

Used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bit field can only be written when the USART is disabled (UE=0).

Note: Bits [7:5] must be kept cleared if Smartcard mode is used.

This bit field is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to Section 22.4: USART implementation.

22.7.6 Receiver timeout register (USARTx_RTOR)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bit-field gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In Smartcard mode, the Block length counter is reset when TXE=0 (TXFE = 0 in case FIFO mode is enabled).

This bit-field can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bit-field gives the Receiver timeout value in terms of number of baud clocks.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the Start Bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on the fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Refer to Section 22.4: USART implementation.

22.7.7 Request register (USARTx_RQR)

Address offset: 0x18

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w_r0	w_r0	w_r0	w_r0	w_r0

Bits 31:5 Reserved, must be kept at reset value

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, Writing 1 to this bit sets the TXE flag.
This allows to discard the transmit data. This bit must be used only in Smartcard mode, when data has not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to ‘0’
When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data is written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO that is, it clears the bit RXFNE.
This allows to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: In the case the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the USART_ISR and request an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to ‘0’. Refer to Section 22.4: USART implementation.

22.7.8 Interrupt and status register (USARTx _ISR)

Address offset: 0x1C

Reset value: 0x00C0 (In case FIFO disabled).

Reset value: 0x28000C0 (In case FIFO/Smartcard mode enabled).

Reset value: 0x08000C0 (In case FIFO enabled/Smartcard mode Disabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	RE ACK	TE ACK	Res.	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE/TX FNF	TC	RXNE/ RXFNE	IDLE	ORE	NF	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the programmed threshold in TXFTCFG in USARTx_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the USART_CR3 register.

0: TXFIFO doesn't reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the programmed threshold in RXFTCFG in USARTx_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the USART_CR3 register.

0: Receive FIFO doesn't reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG threshold is configured to «101», RXFT flag is set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USARTx_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25 **TCBGT:** Transmission complete before guard time flagl

This bit indicates when the last data written in the USART_TDR has been transmitted correctly out of the shift register .

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if there is no NACK from the smartcard. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register. It is cleared by software, writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfuly (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the Smartcard mode, this bit is reserved and forced by hardware to '0'. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is "1".

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when RXFIFO is Full.

An interrupt is generated if the RXFFIE bit =1 in the USART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the USART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering Deepstop mode.

Note: If the USART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wakeup from DeepStop feature, this bit is reserved and forced by hardware to '0'.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character is transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE=1in the USART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_CR3 register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into USARTx_DR.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and forced by hardware to '0

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE=1 in the USART_CR2 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.
The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.
If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 7 **TXE/TXFNF:** Transmit data register empty/TXFIFO not full

When FIFO mode is disabled, TXE is set by hardware when the content of the USARTx_TDR register has been transferred into the shift register. It is cleared by a write to the USARTx_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

When FIFO mode is enabled, TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the USART_TDR. Every write in the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty . After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO. (TXFNF and TXFE is set at the same time).

An interrupt is generated if the TXEIE/TXFNFIE bit =1 in the USART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates when the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware if the transmission of a frame containing data is complete and if TXE/TXFE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXNE/RXFNE**:Read data register not empty/RXFIFO not empty

RXNE bit is set by hardware when the content of the USARTx_RDR shift register has been transferred

to the USARTx_RDR register. It is cleared by a read to the USARTx_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USARTx_RQR register.

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the USART_RDR register. Every read of the USART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXNE/RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE/RXFNEIE=1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).

If mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USARTx_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled) . It is cleared by a software, writing 1 to the ORECF, in the USARTx_ICR register.

An interrupt is generated if RXNEIE/ RXFNEIE=1 or EIE = 1 in the USARTx_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NF**: START bit Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NF flag is set during multi buffer communication if the EIE bit is set.

Note: When the line is noise-free, the NF flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 22.5.6: Tolerance of the USART receiver to clock deviation).

Note: In FIFO mode, this error is associated with the character in the USARTx_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

In Smartcard mode, in transmission, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the USARTx_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: In FIFO mode, this error is associated with the character in the USARTx_RDR.

22.7.9 Interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag
Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bit 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag
Writing 1 to this bit clears the UDRF flag in the USART_ISR register.
Note: If the USART does not support SPI slave mode, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag
Writing 1 to this bit clears the EOBF flag in the USART_ISR register.
Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 22.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NF flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

22.7.10 Receive data register (USART_RDR)

Address offset: 0x24

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 125*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

22.7.11 Transmit data register (USART_TDR)

Address offset: 0x28

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USARTx_TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 125*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.

22.7.12 Prescaler register (USARTx_PRESC)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaing combinations: Reserved.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is «1011» i.e. input clock divided by 256.

22.7.13 USART register map

The table below gives the USART register map and reset values.

Table 108. USART register map and reset values

Offset	Register	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
0x00	USART_CR1	RXFFIE	RXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	Res.	UE
	Reset value	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
0x04	USART_CR2	ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	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
0x08	USART_CR3	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			TCBGTIE	TXFTIE	Res.	Res.	Res.	SCARCNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	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
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	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
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value										0	0		0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	EOBCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value															0					0	0		0	0		0		0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 108. USART register map and reset values (continued)

Offset	Register	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
0x2C	USART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

23 Low power universal asynchronous receiver transmitter (LPUART)

23.1 LPUART introduction

The low power universal asynchronous receiver transmitted (LPUART) is an UART which allows bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to allow UART communications up to 9600 baud/s. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in Deepstop mode , the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex wingle wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memroy access) can be used for data transmission/reception.

23.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud/s to 9600 baud/s using a 32.768 kHz clock source.
- Higher baud rates can be achived by suing a higher frequency clock source
- Two internal FIFOs for transmit and receive data, that can be enabled/disabled by software. FIFOs come with status flags for FIFOs states.
- Dual clock domain allowing
  - UART functionality and wakeup from Deepstop mode
  - Convenient baud rate programming independent from the PCLK reprogramming
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytesare buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications

  The LPUART enters mute mode if the address does not match.
- Wakeup from mute mode (by idle line detection or address mark detection)

*Note: Refer to Section 22.4: USART implementation USART implementation for a comparison of LPUART versus USART featu*res.

23.3 LPUART functional description

Any LPUART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX**: Receive Data Input.

  This is the serial data input.
- **TX:** Transmit Data Output.

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

Through these pins, serial data is transmitted and received in normal LPUART mode as frames comprising:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (7 or 8 or 9 bits) least significant bit first
- 1, 2 stop bits indicating that the frame is complete
- The LPUART interface uses a baud rate generator
- A status register (LPUART_ISR)
- Receive and transmit data registers (LPUART_RDR, LPUART_TDR)
- A baud rate register (LPUART_BRR)

Refer to *Section 23.5: LPUART registers* for the definitions of each bit.

The following pins are required in RS232 Hardware flow control mode:

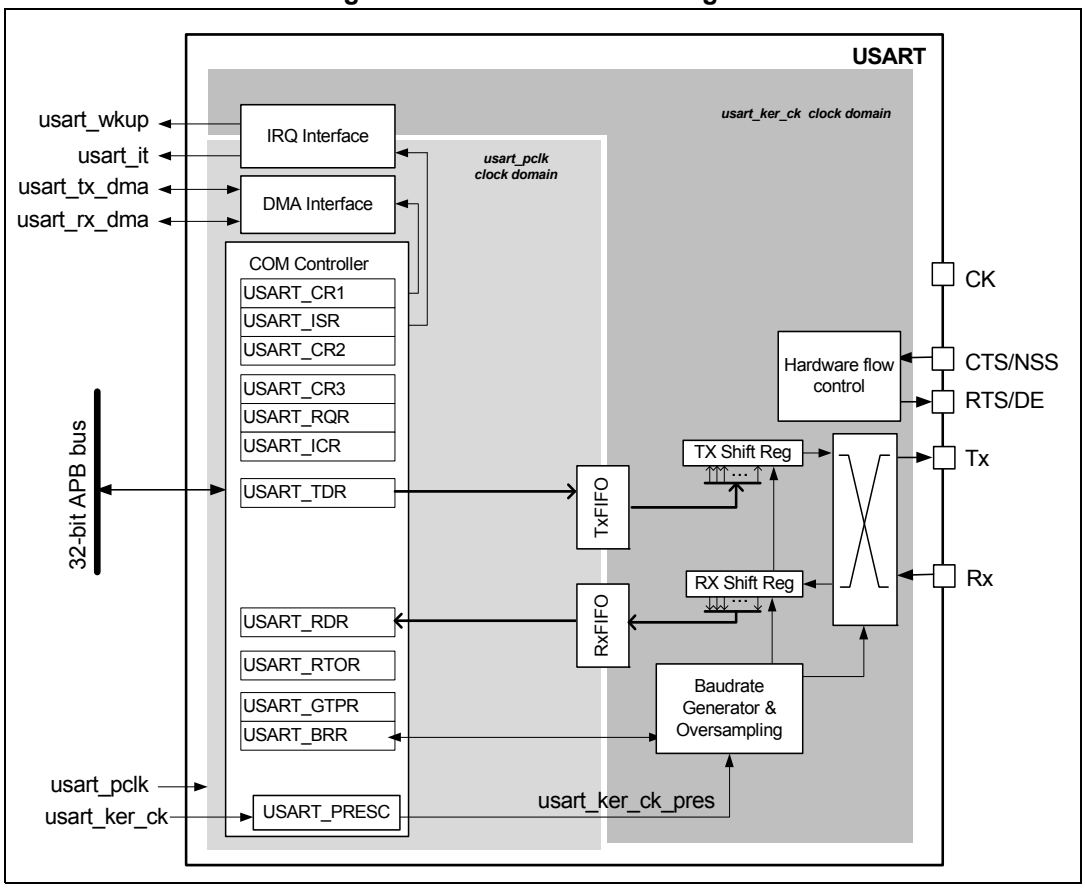
- **nCTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS:** Request to send indicates that the LPUART is ready to receive data (when low).

The following pin is required in RS485 Hardware control mode:

- **DE**: Driver Enable activates the transmission mode of the external transceiver.

Note: DE and nRTS share the same pin.

Figure 150. LPUART Block diagram



The simplified block diagram given in *Figure 150* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The lpuart_pclk clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- The **lpuart_ker_ck** kernel clock domain

  The lpuart_ker_ck is the LPUART clock source. It is independent of the lpuart_pclk and delivered by the RCC. So, the LPUART registers can be written/read even when the lpuart_ker_ck is stopped.

There is no constraint between lpuart_pclk and lpuart_ker_ck: lpuart_ker_ck can be faster or slower than lpuart_pclk, with no more limitation than the ability for the software to manage the communication fast enough.

23.3.1 LPUART character description

Word length may be selected as being either 7 or 8 or 9 bits by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 151*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

In default configuration, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

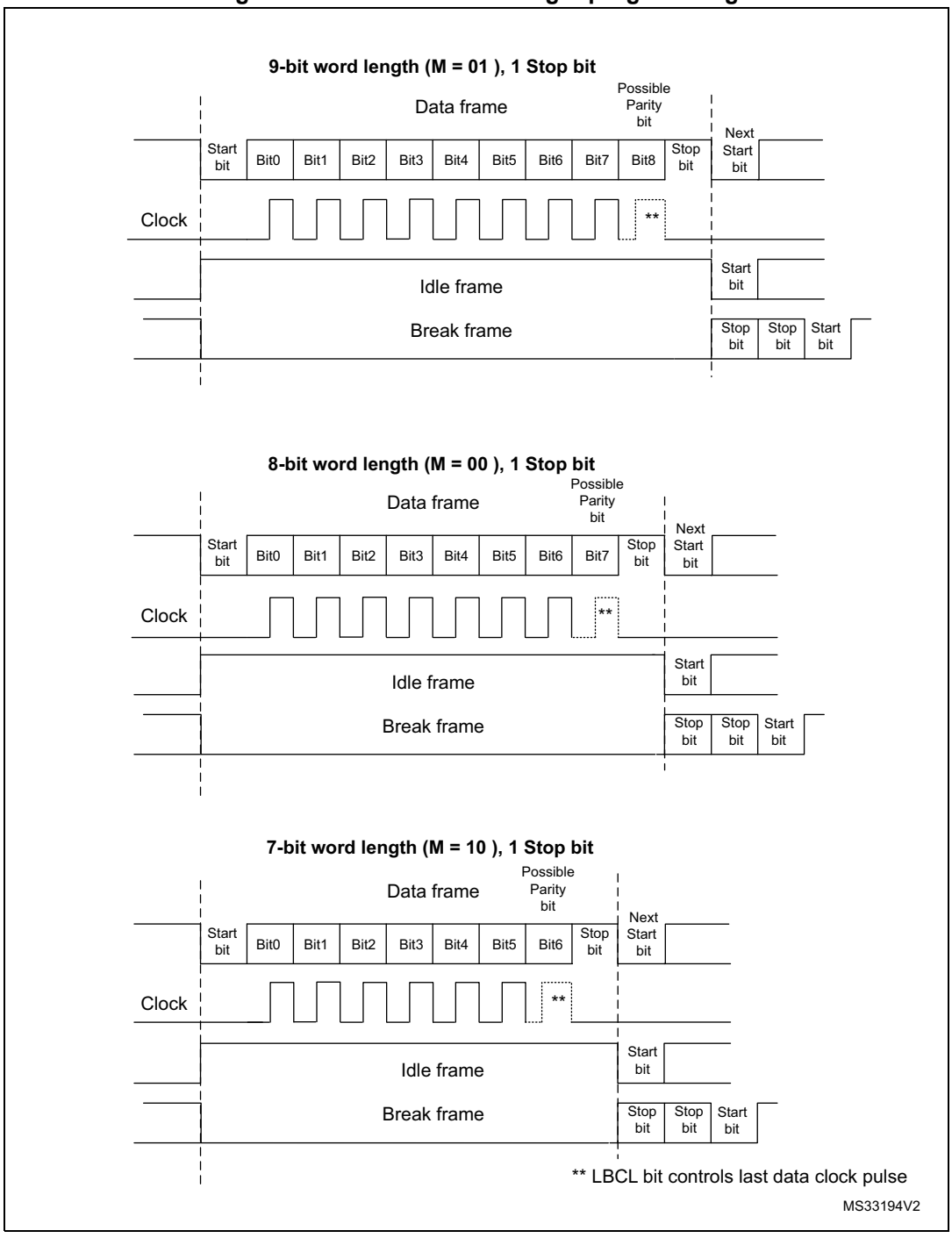
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 151. LPUART Word length programming



23.3.2 FIFOs and thresholds

The LPUARTcan operate in FIFO mode, with the FIFO buffers having a depth of 16 bytes.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO) . The FIFO mode is enabled by setting the bit 29 FIFOEN in USARTx_CR1 register.

Being 9 bits the maximum data word length, the TXFIFO is 9-bits wide. However the RXFIFO is by default 12-bits wide. This is due to the fact that the receiver does not only put the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO with its flags. But If you read RDR, you read just the data. The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupt are triggered. These thresholds are programmed through bit fields RXFTCFG and TXFTCFG in LPUART_CR3 control register.

In this case:

- The receive interrupt in generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.
- The transmit interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

RXFIFO threshold

The RXFIFO threshold is configured using the RXFTCFG bits fields in the LPUART_CR3 register.

When the number of received data is equal to the programmed RXFTCFG, the flag RXFT in the LPUART_ISR register is set.

Having RXFT flag set means that there are RXFTCFG data received: 1 data in LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. So, when the RXFTCFG is programmed to «101» , the RXFT flag is set when 16 data are received: 15 data in the RXFIFO and 1 data in the LPUARTx_RDR. Consequently, the 17th received data does not set the overrun flag.

23.3.3 Transmitter

The transmitter can send data words of either 7 or 8 or 9 bits depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 150*).

When FIFO mode is enabled, data written to the LPUART_TDR register is queued in the TXFIFO.

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by LPUART: 1 and 2 stop bits.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters gets frozen. The current data being transmitted is lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

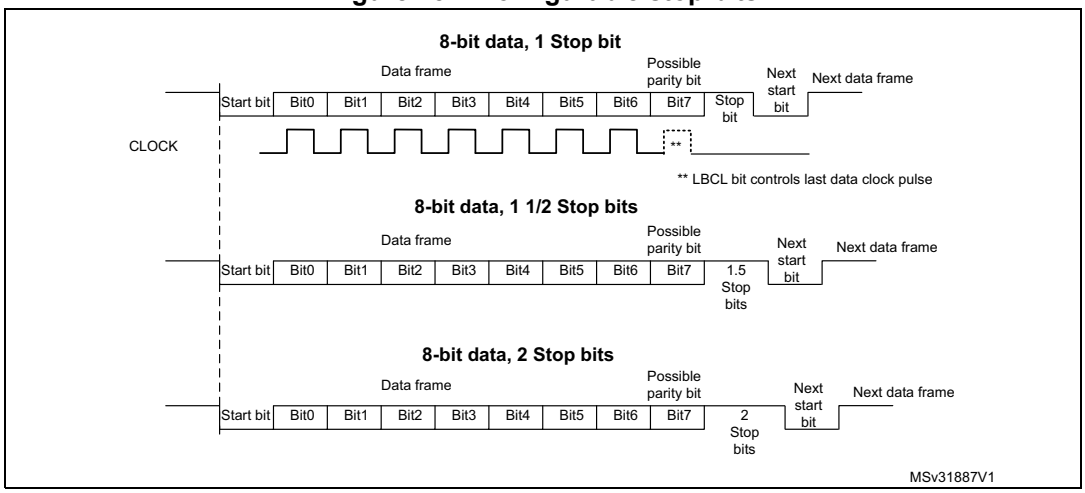
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 152. Configurable stop bits



Character transmission procedure

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO and write operations in the LPUART_TDR are made when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. After writing the last data into the LPUART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the LPUART is disabled or enters the Halt mode to avoid corrupting the last transmission.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.

– When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

Single byte communication

- When FIFO mode is disabled:

Clearing the TXE bit is always performed by a write to the transmit data register.

The TXE flag is set by hardware and it indicates:

– The data has been moved from the LPUART_TDR register to the shift register and the data transmission has started.

– The LPUART_TDR register is empty.

– The next data can be written in the LPUART_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is taking place, a write instruction to the LPUART_TDR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware and it indicates:
  – The TXFIFO is not full.
  – The LPUART_TDR register is empty.
  – The next data can be written in the LPUART_TDR register without overwriting the previous data. When a transmission is taking place, a write operation to the LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO into the shift register at the end of the current transmission.

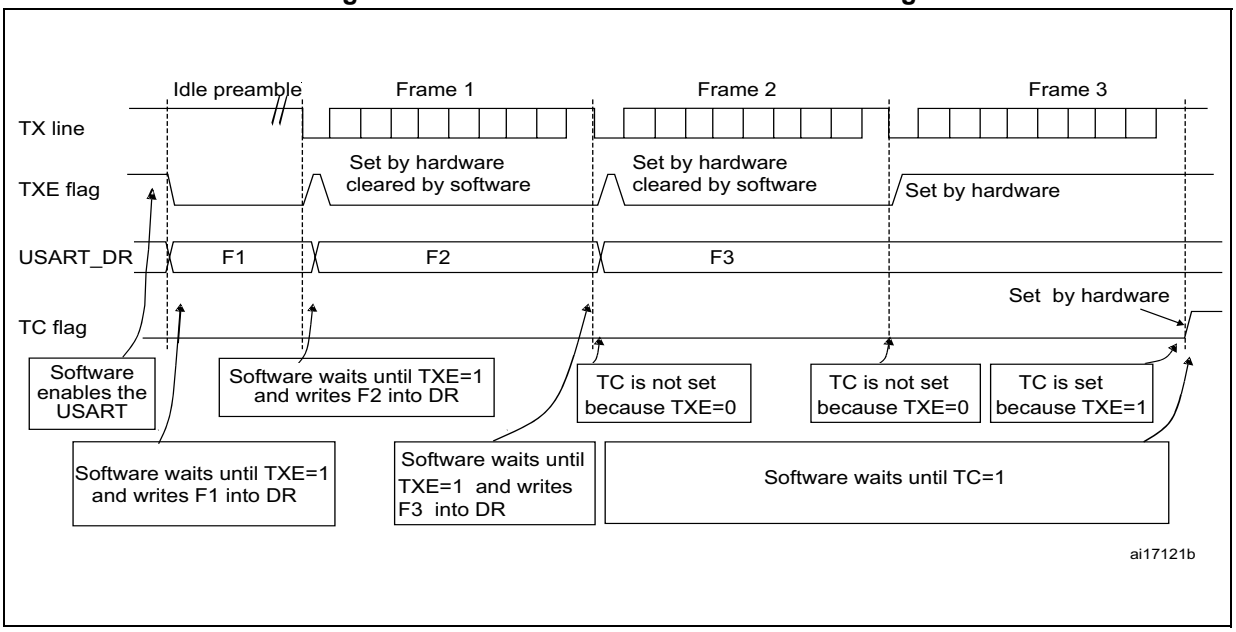
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO isfull. This flag generatesan interrupt if TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written into TXFIFO when the TXFIFO threshold ios reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low power mode (See *Figure 153: TC/TXE behavior when transmitting*).

Figure 153. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 151*).

If a ‘1’ is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

23.3.4 Receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In LPUART, for START bit detection, a falling edge should be detected first on the Rx line, then a sample is taken in the middle of the start bit to confirm that it is still ‘0’. If the start sample is at ‘1’, then the noise error flag (NF) is set, then the START bit is discarded and the

receiver waits for a new START bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data shifts in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the RE bit LPUART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. A read of LPUART_RDR gets the oldest entry in the RXFIFO. When a data is received, it is stored in the RXFIFO, with error bits associated with that data.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE bit is performed by a software read to the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read of LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is

reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as for a received data character plus an interrupt if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled: An overrun error occurs when a character is received when RXNE has not been reset. data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - The ORE bit is set.
  - The RDR content is not lost. The previous data is available when a read to
  - LPUART_RDR is performed.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXNEIE bit is set or EIE bit is set.
- FIFO mode enabled: an overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full. Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty. An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - The ORE bit is set.
  - The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE bit is set or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

In case of FIFO mode is disabled, there are two possibilities

- *if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,*
- *if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the Clock Control system (see the *Section 6: Reset and clock controller (RCC)*). The clock source must be chosen before enabling the LPUART (by setting the UE bit).

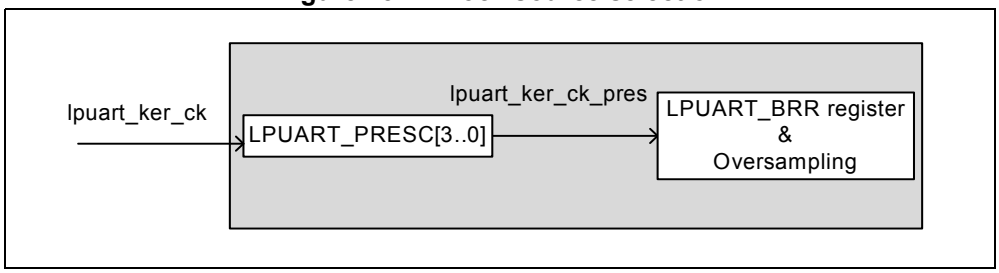
The choice of the clock source must be done according to two criteria:

- Possible use of the LPUART in low power mode
- Communication speed.

The clock source frequency is lpuart_kern_ck.

When the dual clock domain and the wakeup from Deepstop mode features are supported, the lpuart_kern_ck clock source can be selected through *Section 6.6.2: Clock configuration register (RCC_CFGR)*. The lpuart_kern_ck can be divided by a programmable factor in the LPUARTx_PRESC register.

Figure 154. CLock source selection



Choosing lpuart_kern_ck as LSE clock source may allow the LPUART to receive data while the MCU is in low power mode. Depending on the received data and wakeup mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active in order to allow LPUART communication.

When the LPUART clock source is configured to be LSE, it is possible to keep enabled this clock during Deepstop mode by setting the UCESM bit in LPUART_CR3 control register.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming baud as close as possible to the middle of the baud-period. Only a single sample is taken of each of the incoming bauds.

Note: There is no noise detection for data.

Framing error

A framing error is detected when:

The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode.

- ***1 stop bit***: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
- ***2 stop bits***: Sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

23.3.5 Baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times f_{CKPRES}}{\text{LPUARTDIV}}$$

LPUARTDIV is coded on the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value should not be changed during communication.

It is forbidden to write values less than 0x300 in the LPUART_BRR register.

fck must be in the range [3 x baudrate ..4096 x baudrate]

Table 109. Error calculation for programmed baudrates at f_{ck} = 32,768 KHz

Baud rate		f_{CK} = 32,768 KHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	300 Bps	300 Bps	0x6D3A	0
2	600 Bps	600 Bps	0x369D	0
3	1200 Bps	1200.087 Bps	0x1B4E	0.007
4	2400 Bps	2400.17 Bps	0xDA7	0.007
5	4800 Bps	4801.72 Bps	0x6D3	0.035
6	9600 Bps	9608.94 Bps	0x369	0.093

Table 110. Error calculation for programmed baudrates at f_{ck} = 16 MHz

Baud rate		f_{CK} = 16 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	9.6 kBps	9.60001 kBps	0x682AA	0
2	19.2 kBps	19.20003 kBps	0x34155	0
3	38.4 kBps	38.40024 kBps	0x1A0AA	0
4	57.6 kBps	57.60009 kBps	0x115C7	0
5	115.2 kBps	115.2018 kBps	0x8AE3	0.001
6	230.4 kBps	230.41008 kBps	0x4571	0.004
7	460.8 kBps	460.84608 kBps	0x22B8	0.01
8	921.6 kBps	921.69216 kBps	0x115C	0.01
9	1843.2 kBps	1843.38433 kBps	0x8AE	0.01
10	3686.4 kBps	3686.76867 kBps	0x457	0.01

23.3.6 Multiprocessor communication

It is possible to perform multiprocessor communication with the LPUART (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In order to use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two UCLK cycles) otherwise mute mode might remain active.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

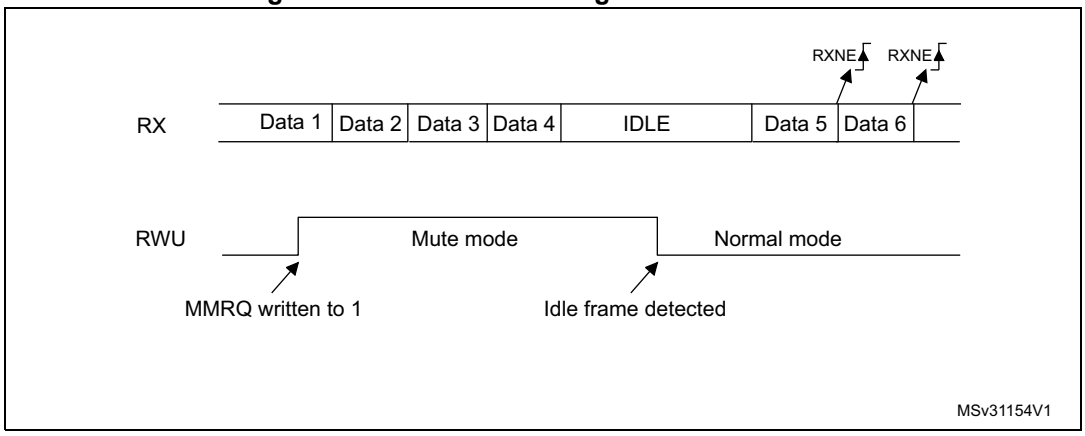
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 155*.

Figure 155. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-

bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

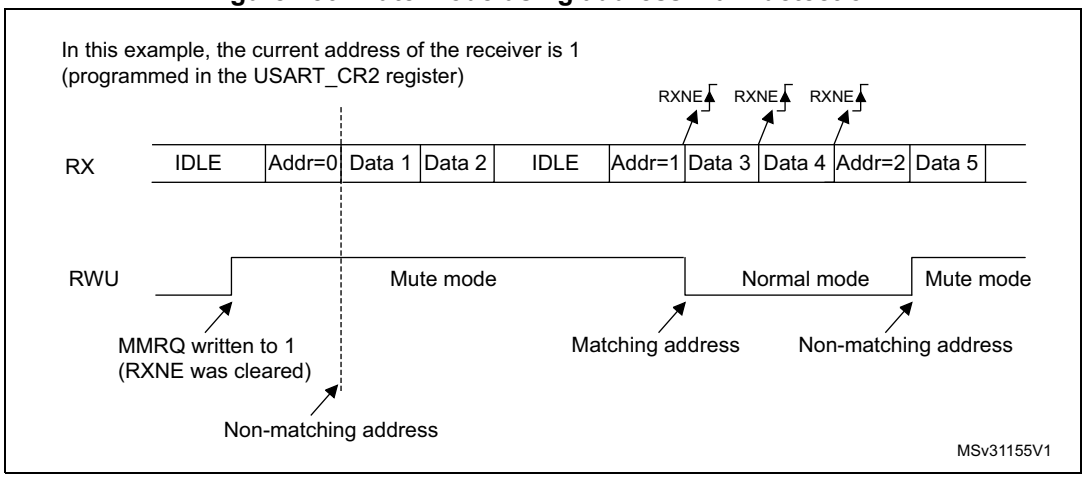
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 156*.

Figure 156. Mute mode using address mark detection



23.3.7 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 111*.

Table 111. Frame formats

Table 112:

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |

Table 112:

M bits	PCE bit	LPUART frame(1)
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

23.3.8 Single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the LPUART_CR2 register,
- SCEN and IREN bits in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflicts on the line must be managed by software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART, in the case of 1-STOP bit configuration, the RXNE flag is set in the middle of the STOP bit.

23.3.9 Continuous communication using DMA

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 23.4: LPUART interrupts to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 23.3.4. To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

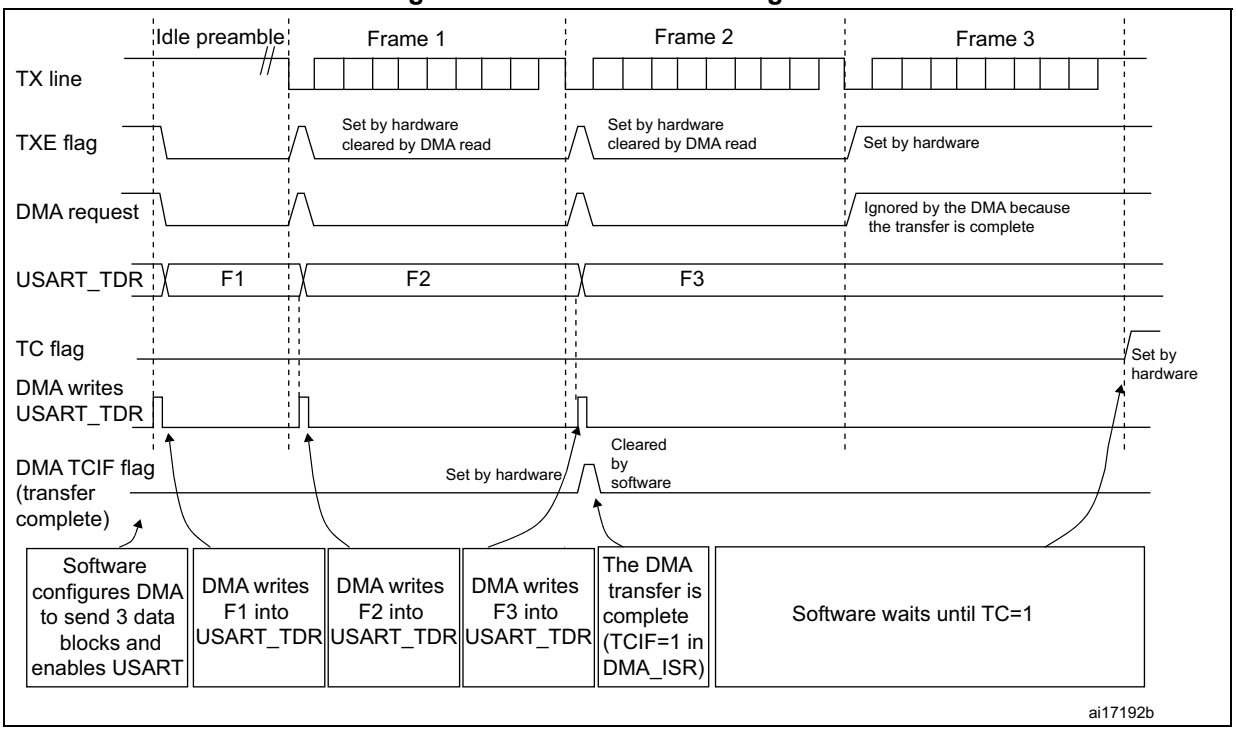
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to *Section 10: DMA controller (DMA)*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering Deepstop mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 157. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

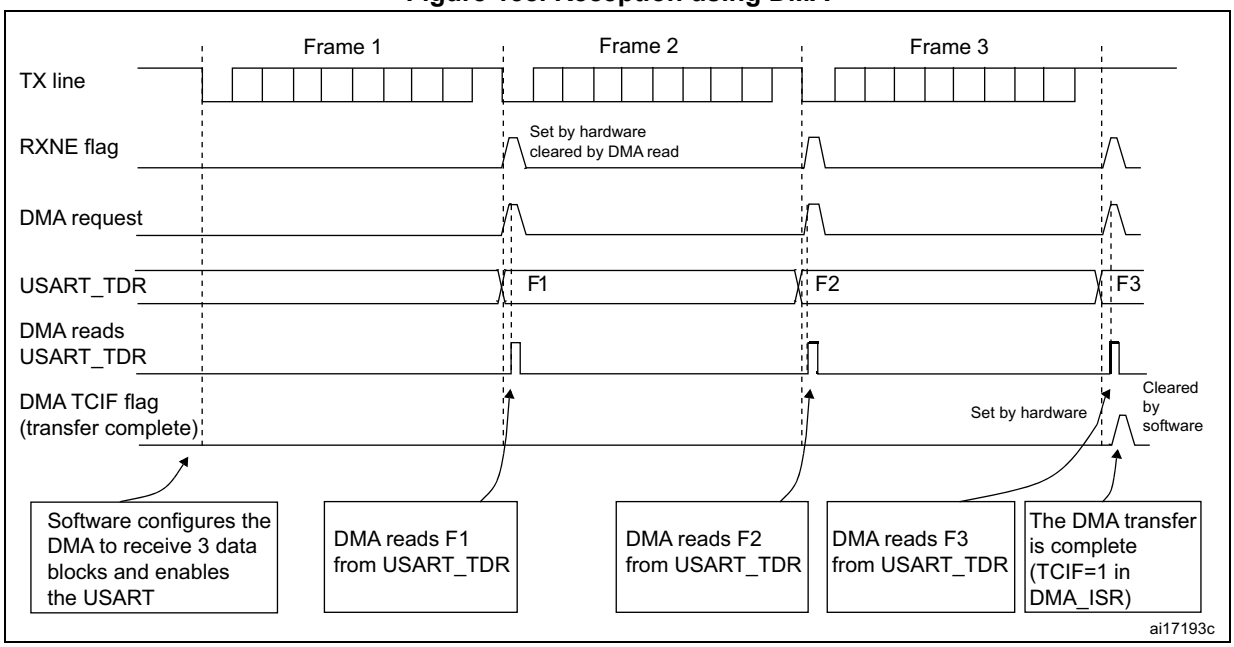
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data is loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to *Section 10: DMA controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 158. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

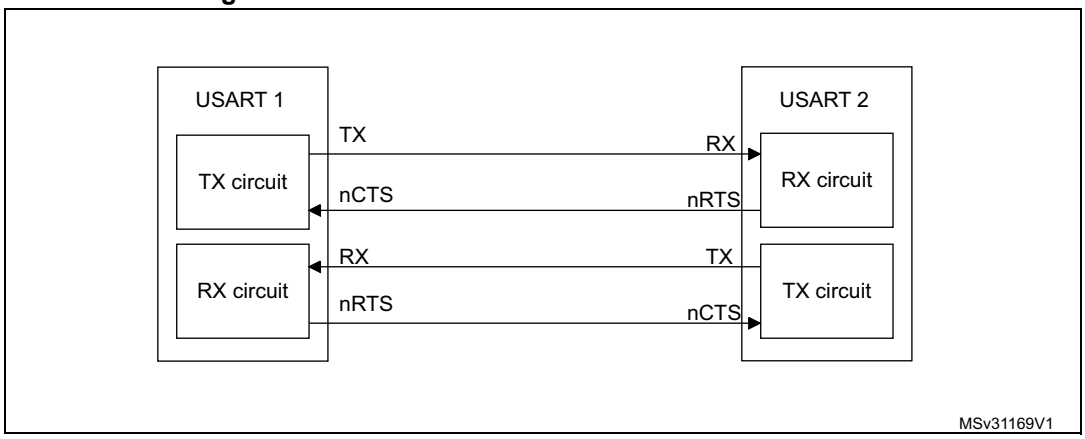
Error flagging and interrupt generation in multibuffer communication

In multibuffer communication if any error occurs during the transaction the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

23.3.10 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. *Figure 159* shows how to connect 2 devices in this mode:

Figure 159. Hardware flow control between 2 LPUARTs

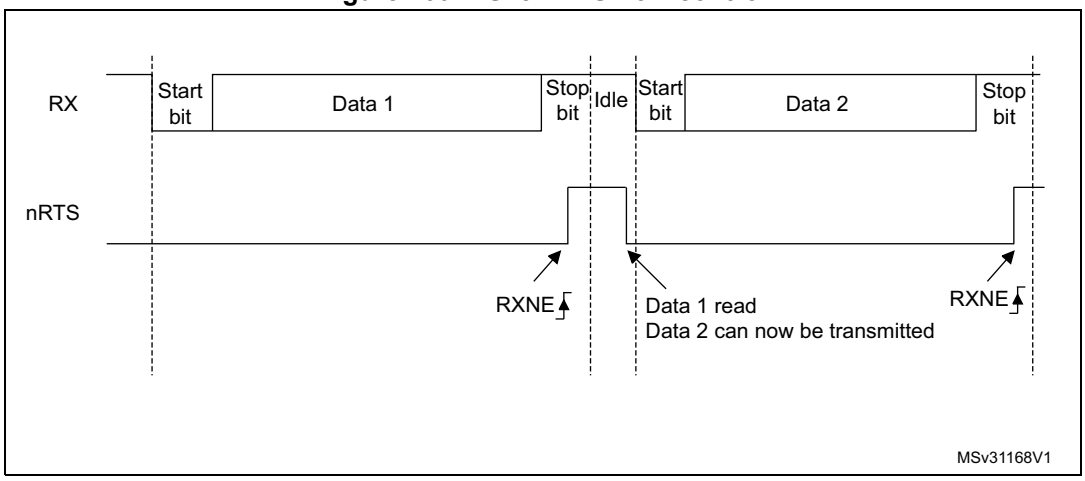


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 160* shows an example of communication with RTS flow control enabled.

Figure 160. RS232 RTS flow control



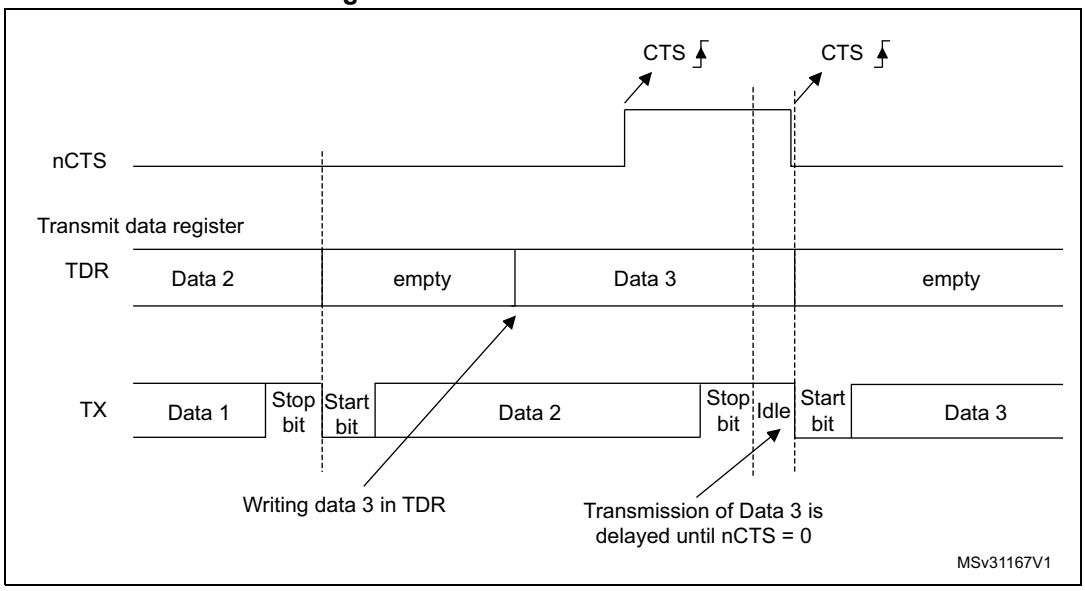
Note: When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 161* shows an example of communication with CTS flow control enabled.

Figure 161. RS232 CTS flow control



Note: For correct behavior, nCTS must be asserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 Driver Enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the START bit. It is programmed using the DEAT [4:0] bit fields in the LPUART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bit fields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

In LPUART, the DEAT and DEDT are expressed in LPUART clock source (f_{CK}) cycles:

- The Driver enable assertion time =
  - (1 + (DEAT x P)) x f_{CK} , if P <> 0
  - (1 + DEAT) x f_{CK} , if P = 0
- The Driver enable de-assertion time =
  - (1 + (DEDT x P)) x f_{CK} , if P <> 0
  - (1 + DEDT) x f_{CK} , if P = 0

with P = BRR[20:11]

23.3.11 Wakeup from Deepstop mode

The LPUART is able to wake up the MCU from Deepstop mode when the UESM bit is set and the LPUART clock is set to LSE (see *Section 6: Reset and clock controller (RCC)*).

When FIFO mode is disabled, the MCU wakeup from Deepstop mode can be done using the standard RXNE interrupt. In this case, the RXNEIE bit must be set before entering Deepstop mode.

When FIFO mode is enabled, the MCU wakeup from Deepstop mode can be done using the:

- standard RXFIFO not empty interrupt. In this case, the RXFNEIE bit must be set before entering Deepstop mode.
- standard RXFIFO full interrupt. In this case, the RXFFIE bit must be set before entering Deepstop mode.
- standard TXFIFO empty interrupt. In this case, the TXFEIE bit must be set before entering Deepstop mode. This allows sending the data in the TXFIFO during Deepstop mode. When all data are sent (i.e. TXFIFO is empty), the MCU wakes up from Deepstop mode.

In order to avoid overrun/underrun errors and transmit/receive data in Deepstop mode, the MCU wakeup from Deepstop mode can be done also using the:

- standard TXFIFO threshold interrupt.
- standard RXFIFO threshold interrupt.

An application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from Deepstop mode allows doing as many USART transfers as possible during Deepstop mode with the benefit of optimizing consumption.

Alternatively, a specific interrupt may be selected through the WUS bit fields.

In order to be able to wake up the MCU from Deepstop mode, the UESM bit in the USART_CR1 control register must be set prior to entering Deepstop mode.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUFIE bit is set.

Note: Before entering Deepstop mode, the user must ensure that the USART is not performing a transfer. BUSY flag cannot ensure that Deepstop mode is never entered during a running reception.

Note: The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in Deepstop or in an active mode.

Note: When entering Deepstop mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the USART is actually enabled.

Note: When DMA is used for reception, it must be disabled before entering Deepstop mode and re-enabled upon exit from Deepstop mode.

Note: The wakeup from Deepstop mode feature is not available for all modes. For example it doesn't work in SPI mode because the SPI operates in master mode only.

Note: When FIFO is enabled, the wakeup from Deepstop mode on address match is only possible when mute mode is enabled.

Using Mute mode with Deepstop mode

If the USART is put into Mute mode before entering Deepstop mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in Deepstop mode.
- If the wakeup from Mute mode on address match is used, then the source of wakeup from Deepstop mode must also be the address match. If the RXNE flag is set when entering the Deepstop mode, the interface remains in mute mode upon address match and wake up from STOP.

Note: When FIFO management is enabled,mute mode is used with wakeup from Deepstop mode without any constraints (i.e.the two points mentioned above about mute and Deepstop mode are valid only when FIFO management is disabled).

23.4 LPUART interrupts

Table 113. LPUART interrupt requests

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				lpuart_it	lpuart_wkup
Transmit data register empty	TXE	TXEIE	TXE cleared when a data is written in TDR	YES	NO
Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFNF cleared when TXFIFO is full.	YES	NO
Transmit FIFO Empty	TXFE	TXFEIE	TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.	YES	YES
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT cleared by hardware when the TXTFIFO content is less than programmed threshold	YES	YES
CTS interrupt	CTSIF	CTSIE	CTSIF cleared by software by setting CTSCF bit.	YES	NO
Transmission Complete	TC	TCIE	TC cleared when a data is written in TDR or by setting TCCF bit.	YES	NO
Receive data register not empty (data ready to be read)	RXNE	RXNEIE	RXNE cleared by reading RDR or by setting RXFRQ bit.	YES	YES
Receive FIFO Not Empty	RXFNE	RXFNEIE	RXFNE cleared when the RXFIFO is emty or by setting RXFRQ bit.	YES	YES
Receive FIFO Full	RXFF(1)	RXFFIE	RXFF cleared when the RXFIFO contains at least one data.	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT cleared by hardware when the RXFIFO content is less than programmed threshold	YES	YES

Table 113. LPUART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				lpuart_it	lpuart_wkup
Overrun error detected	ORE	RX-NEIE/RX-FNEIE	ORE cleared by setting ORECF bit.	YES	NO
Idle line detected	IDLE	IDLEIE	IDLE cleared by setting IDLECF bit.	YES	NO
Parity error	PE	PEIE	PE cleared by setting PECF bit.	YES	NO
Noise error, Overrun error and Framing Error in multi-buffer communication.	NE or ORE or FE	EIE	NE cleared by setting NECF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.	YES	NO
Character match	CMF	CMIE	CMF cleared by setting CMCF bit.	YES	NO
Wakeup from low power mode	WUF(2)	WUFIE	WUF is cleared by setting WUCF bit.	YES	YES

1. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In DEESTOP mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
2. The WUF interrupt is active only in low power mode

23.5 LPUART registers

Refer to *Section 1.5 on page 35* for a list of abbreviations used in register descriptions.

23.5.1 Control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE TXFNFIE	TCIE	RXNEIE RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE** :RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF=1 in the LPUART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 30 **TXFEIE** :TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE=1 in the LPUART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 29 **FIFOEN** :FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length
This bit, with bit 12 (M0) determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 Start bit, 8 Data bits, n Stop bit
M[1:0] = 01: 1 Start bit, 9 Data bits, n Stop bit
M[1:0] = 10: 1 Start bit, 7 Data bits, n Stop bit
This bit can only be written when the LPUART is disabled (UE=0).

Note: In 7-bits data length mode, the Smarcard mode, LIN master mode and Autobaudrate (0x7F and 0x55 frames detection) are not supported.

Bit 27 Reserved, must be kept at reset value

Bit 26 Reserved, must be kept at reset value

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in UCLK (LPUART clock) clock cycles. For more details, refer to *Section 23.3.10: RS232 Hardware flow control and RS485 Driver Enable*.

This bit field can only be written when the LPUART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal..It is expressed in UCLK (LPUART clock) clock cycles. For more details, refer to *Section 23.3.10: RS232 Hardware flow control and RS485 Driver Enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 15 Reserved, must be kept at reset value

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit, with bit 28 (M1) determine the word length. It is set or cleared by software. See Bit 28 (M1)description.

This bit can only be written when the LPUART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bit field can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXEIE/TXFNFIE**: Transmit data regsiter empty/TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXNEIE/RXFNEIE**: Receive data register not empty/RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Note: During transmission, a “0” pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.
When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bits 1 **UESM**: LPUART enable in Deepstop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from Deepstop mode.

When this bit is set, the LPUART is able to wake up the MCU from Deepstop mode, provided that the LPUART clock selection is LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from Deepstop mode.

1: LPUART able to wake up the MCU from Deepstop mode. When this function is active, the clock source for the LPUART must be LSE (see *Section 6: Reset and clock controller (RCC)*)

Note: It is recommended to set the UESM bit just before entering Deepstop mode and clear it on exit from Deepstop mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low power mode

1: LPUART enabled

Note: In order to go into low power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

23.5.2 Control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:28 **ADD[7:4]**: Address of the LPUART node

This bit-field gives the address of the LPUART node or a character code to be recognized.
This is used in multiprocessor communication during Mute mode or Deepstop mode, for wakeup with 7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. It may also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.
This bit field can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the LPUART node

This bit-field gives the address of the LPUART node or a character code to be recognized.
This is used in multiprocessor communication during Mute mode or Deepstop mode, for wakeup with address mark detection.
This bit field can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

Bit 23:20 Reserved, must be kept at reset value

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bit field can only be written when the LPUART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bit field can only be written when the LPUART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: TX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the TX line.
This bit field can only be written when the LPUART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: RX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the RX line.
This bit field can only be written when the LPUART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART.
This bit field can only be written when the LPUART is disabled (UE=0).

Bit 14 Reserved, must be kept at reset value

Bits 13:12 **STOP[1:0]**: STOP bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bit field can only be written when the LPUART is disabled (UE=0).

Bit 11:5 Reserved, must be kept at reset value

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

23.5.3 Control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG			RXFTIE.	RXFTCFG			Res.	TXFTIE	WUFIE	WUS[2:0]		Res.	Res.	Res.	Res.
rw			rw	rw				rw	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
010:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Remaining combinations: Reserved.

Bit28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
010:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Remaining combinations: Reserved.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from Deepstop mode interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever WUF=1 in the LPUART_ISR register

Note: WUFIE must be set before entering in Deepstop mode.
The WUF interrupt is active only in Deepstop mode.
If the LPUART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bit 21:20 **WUS[1:0]**: Wakeup from Deepstop mode interrupt flag selection
This bit-field specify the event which activates the WUF (Wakeup from Deepstop mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01:Reserved.
10: WUF active on Start bit detection
11: WUF active on RXNE.
This bit field can only be written when the LPUART is disabled (UE=0).

Note: If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode
This bit allows the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on Reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE=0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 : Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE=0).
Note: This control bit allows checking the communication flow w/o reading the data.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.
This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bit 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected
This bit can only be written when the LPUART is disabled (UE=0).

Bit 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the LPUART_ISR register.

23.5.4 Baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]**

Note: It is forbidden to write values less than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be > = 0x300 and LPUART_BRR is 20 bits, a care should be taken when generating high baudrates using high fck values. fck must be in the range [3 x baudrate ..4096 x baudrate].

23.5.5 Request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO . This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data is written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This allows to discard the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: In the case the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value

23.5.6 Interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x00C0 (In case FIFO disabled)

Reset value: 0x08000C0 (In case FIFO enabled)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the programmed threshold in TXFTCFG in LPUARTx_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO doesn't reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the programmed threshold in RXFTCFG in LPUARTx_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the LPUART_CR3 register.

0: Receive FIFO doesn't reach the programmed threshold.

1: ReceiveFIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when RXFIFO is Full.

An interrupt is generated if the RXFFIE bit =1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering Deepstop mode.

Note: If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from Deepstop mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bit field. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

The WUF interrupt is active only in Deepstop mode.

If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_RQR register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character is transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1in the LPUART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception on going

Bit 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE/TXFNF:** Transmit data register empty/TXFIFO not full

When FIFO mode is disabled, TXE is set by hardware when the content of the LPUARTx_TDR register has been transferred into the shift register. It is cleared by a write to the LPUARTx_TDR register.

When FIFO mode is enabled, TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty . After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO. (TXFNF and TXFE is set at the same time).

An interrupt is generated if the TXEIE/TXFNFIE bit =1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE/TXFF is set. An interrupt is generated if TCIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the LPUART_TDR register.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXNE/RXFNE**:Read data register not empty/RXFIFO not empty

RXNE bit is set by hardware when the content of the LPUARTx_RDR shift register has been transferred

to the LPUARTx_RDR register. It is cleared by a read to the LPUARTx_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUARTx_RQR register.

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXNE/RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE/RXFNEIE=1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).

If mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUARTx_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled) . It is cleared by a software, writing 1 to the ORECF, in the LPUARTx_ICR register.

An interrupt is generated if RXNEIE/ RXFNEIE=1 or EIE = 1 in the LPUARTx_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NF**: START bit Noise detection flag

This bit is set by hardware when noise is detected on the START bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NF flag is set during multi buffer communication if the EIE bit is set.

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

In Smartcard mode, in transmission, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

23.5.7 Interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w_r0			w_r0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
						w_r0			w_r0		w_r0	w_r0	w_r0	w_r0	w_r0

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from Deepstop mode clear flag
Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.
Note: If the LPUART does not support the wakeup from Deepstop feature, this bit is reserved and forced by hardware to '0'.

Bit 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag
Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bit 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8:7 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NCF**: Noise detected clear flag
Writing 1 to this bit clears the NF flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

23.5.8 Receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 150*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

23.5.9 Transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 150*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.

23.5.10 Prescaler register (LPUART_PRESC)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaing combinations: Reserved.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is «1011» i.e. input clock divided by 256.

23.5.11 LPUART register map

The table below gives the LPUART register map and reset values.

Table 114. LPUART register map and reset values

Offset	Register	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
0x00	LPUART_CR1	Res.	Res.	Res.	M1	Res.	Res.	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	Res.	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value				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
0x04	LPUART_CR2	ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		WUFIE	WUS [1:0]		Res.	Res.	Res.	Res.	DEP	DEM	DDRE		Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value										0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	Res.	ORECF	NCF	FECF	PECF
	Reset value												0			0								0			0			0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Refer to *Section 2.2.2 on page 44* for the register boundary addresses.

24 Serial peripheral interface / inter-IC sound (SPI/I2S)

24.1 Introduction

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I^2S audio protocol. SPI or I^2S mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

The Inter-IC sound (I^2S) protocol is also a synchronous serial communication interface.It can operate in slave or master mode with half-duplex communication. It can address four different audio standards including the Philips I^2S standard, the MSB- and LSB-justified standards and the PCM standard.

24.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4-bit to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to $f_{PCLK}/2$.
- Slave mode frequency up to $f_{PCLK}/2$.
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- SPI TI mode support

24.3 I2S main features

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave) and Frame Error Flag in reception and transmitter mode (slave only)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-Justified standard (Left-Justified)
  - LSB-Justified standard (Right-Justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock can be output to drive an external audio component. Ratio is fixed at 256 × F_S (where F_S is the audio sampling frequency)

24.4 SPI/I2S implementation

This manual describes the full set of features implemented in SPI3.

The *Table 115* describes the SPI/I2S implementation in the STM32WB09xE device.

Table 115. SPI implementation(1)

SPI Features	SPI3
Hardware CRC calculation	X
Rx/Tx FIFO	X
NSS pulse mode	X
I2S mode	X
TI mode	X

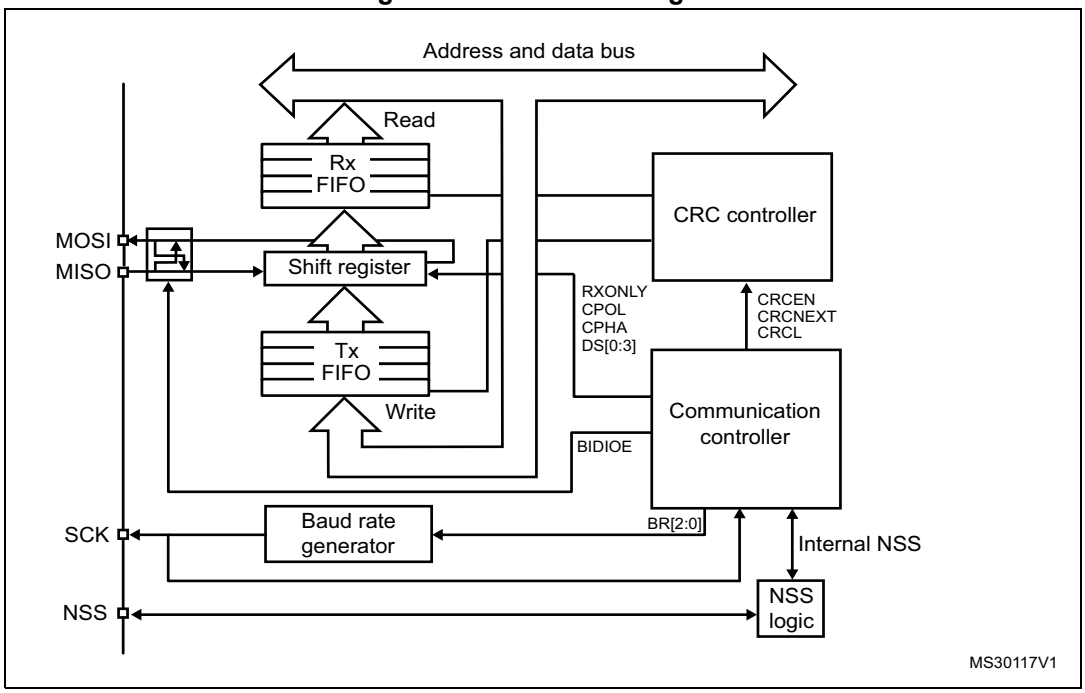
1. X = supported.

24.5 SPI functional description

24.5.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram *Figure 162*.

Figure 162. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.

Note: If the SPI is in master mode and the internal pull-up/-down of the pad is used, the software must take care to activate the pull polarity (up or down) of the I/O to be coherent with the CPOL programming (pull-down if CPOL=0 and pull-up if CPOL=1).

- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

  See *Section 24.5.4: Slave select (NSS) pin management* for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for

synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

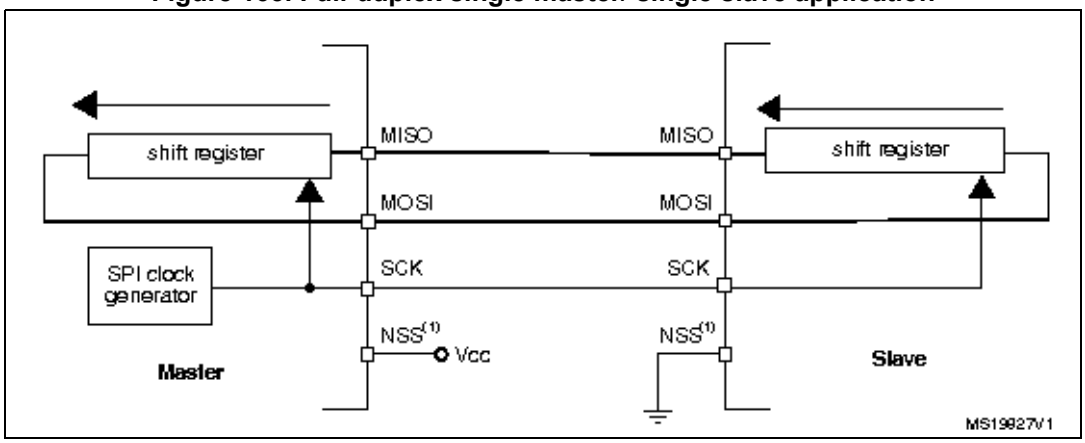
24.5.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 163. Full-duplex single master/ single slave application

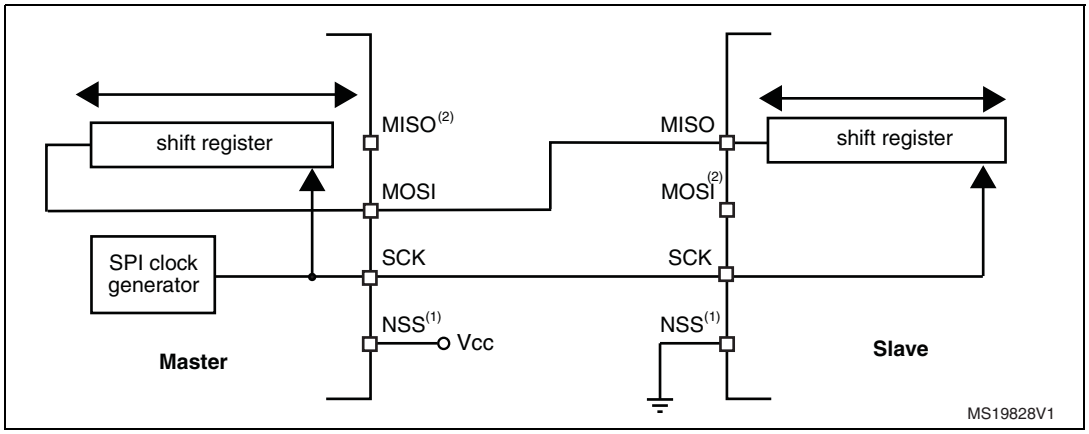


1. The NSS pin is configured as an input in this case.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 164. Half-duplex single master/ single slave application



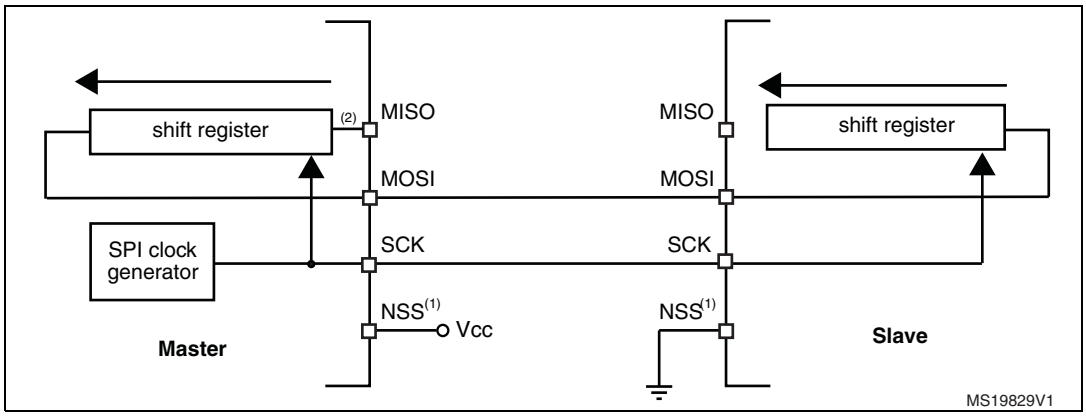
1. The NSS pin is configured as an input in this case.
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1)**: The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *24.5.4: Slave select (NSS) pin management*). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

Figure 165. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)



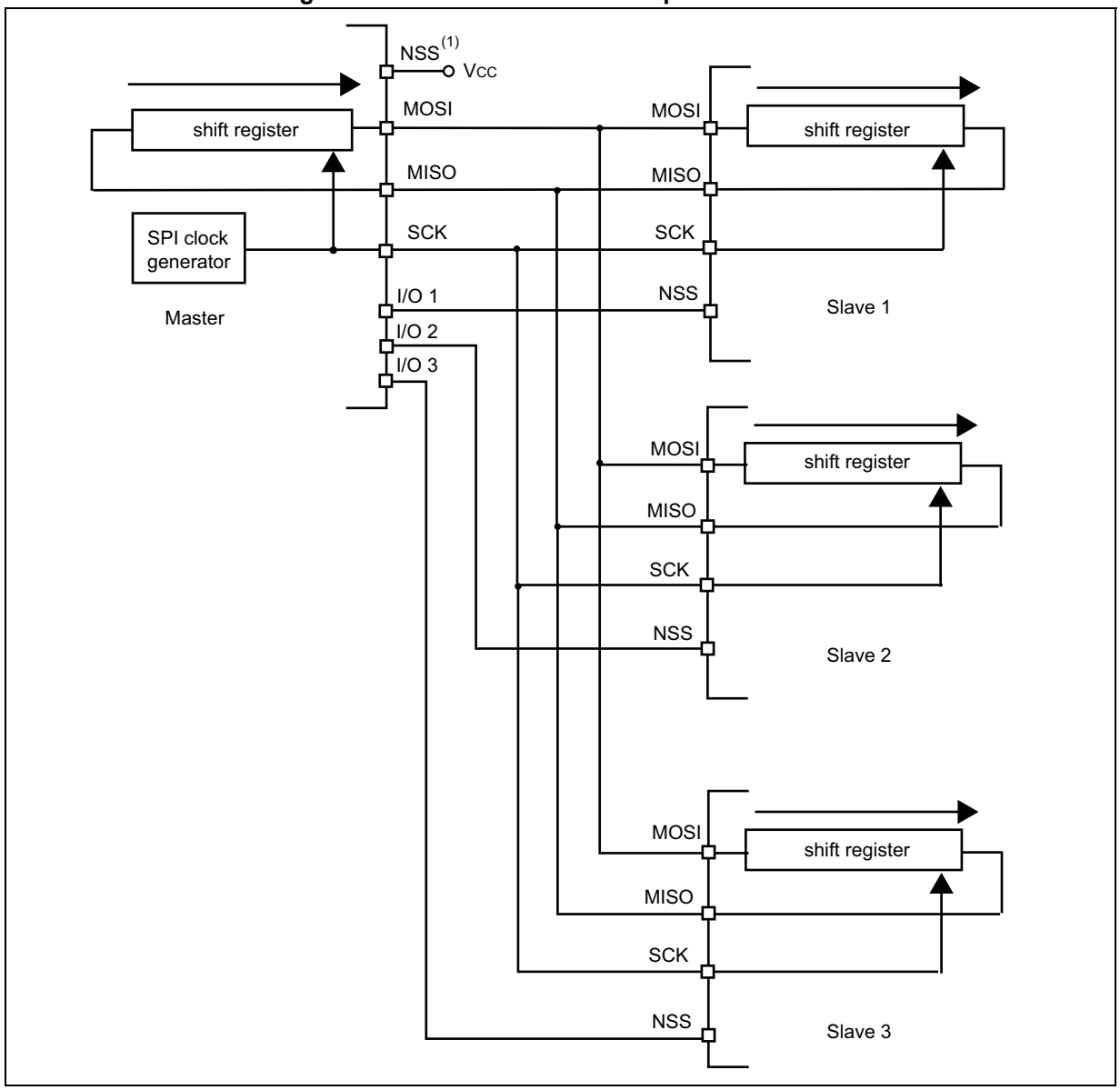
1. The NSS pin is configured as an input in this case.
2. The input information is captured in the shift register and must be ignored in standard transmit only mode (for example, OVF flag)
3. In this configuration, both the MISO pins can be used as GPIOs.

Note: Any simplex communication can be alternatively replaced by a variant of the half duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).

24.5.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see *Figure 166.*). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.

Figure 166. Master and three independent slaves



1. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see *Table 7: Alternate modes 0, 1 and 2* and *Table 8: Alternate modes 3, 4, and 6*).

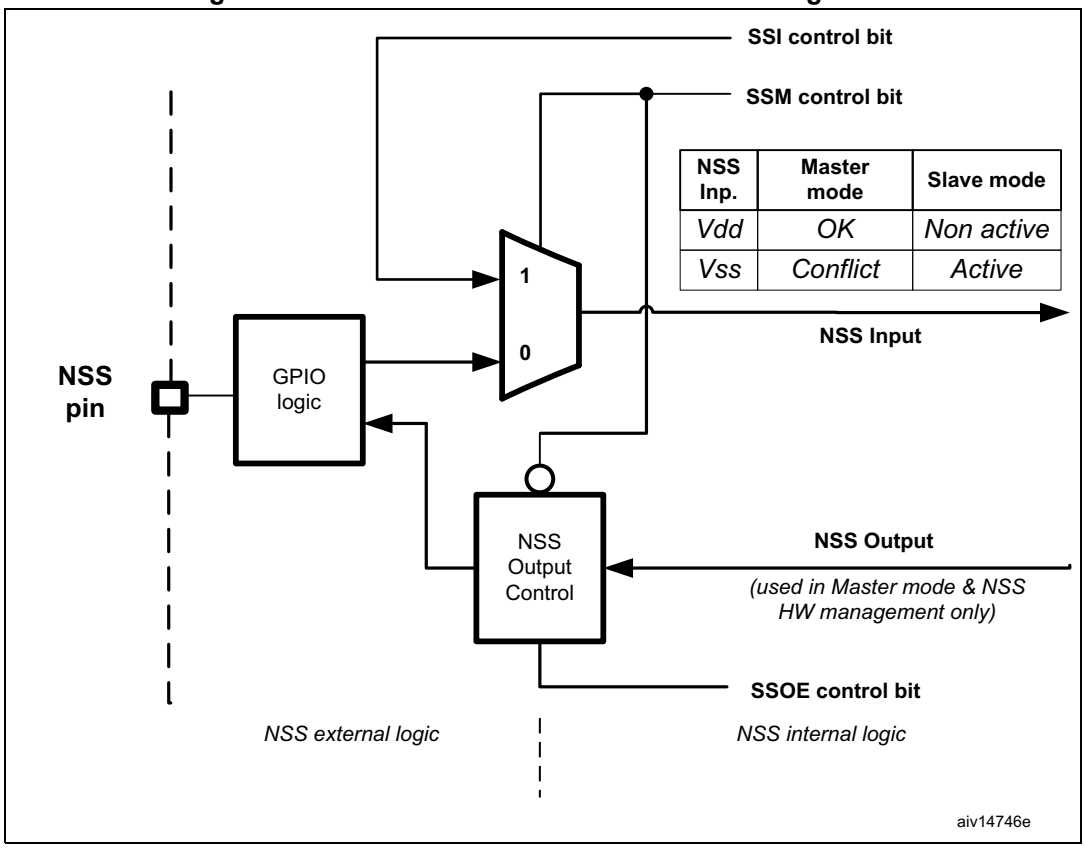
24.5.4 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- **Software NSS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx_CR1).
  - **NSS output enable (SSM=0,SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  - **NSS output disable (SSM=0, SSOE = 0)**: if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 167. Hardware/software slave select management



24.5.5 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

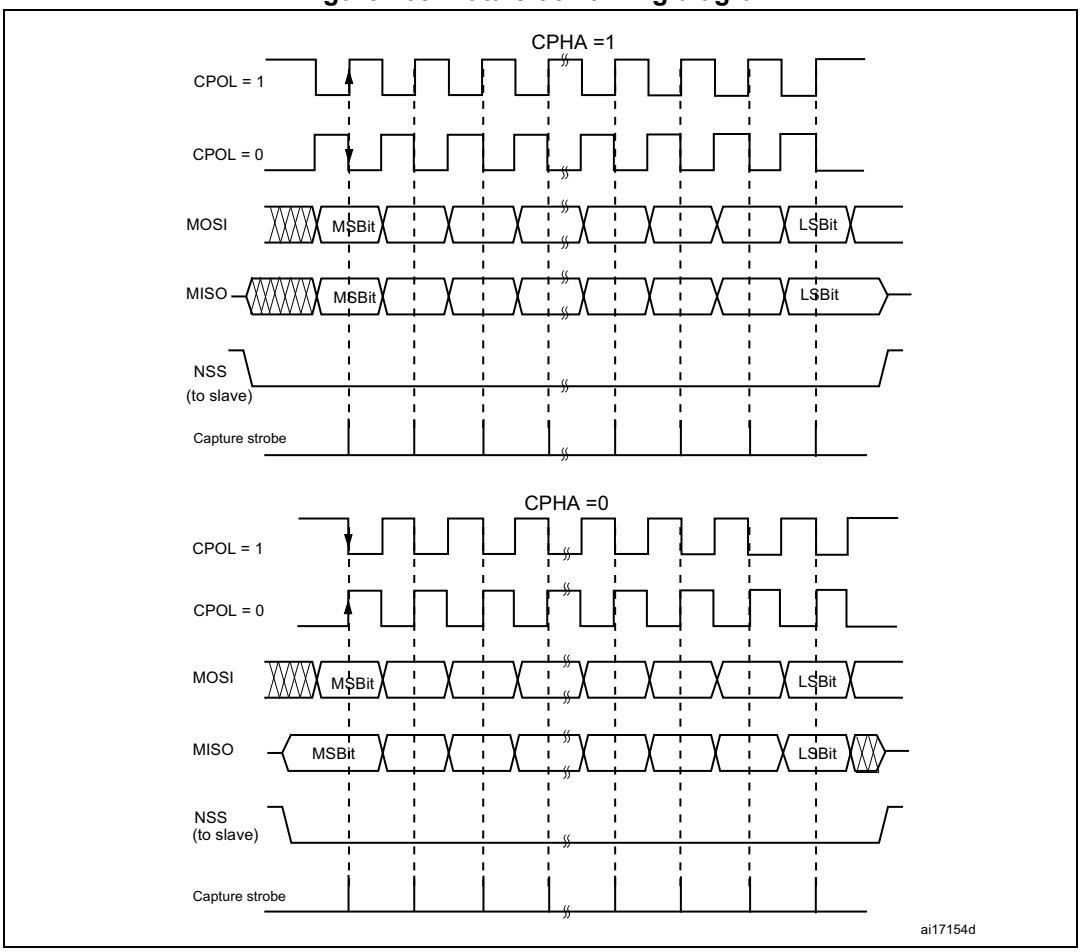
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 168, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPIx_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

Figure 168. Data clock timing diagram

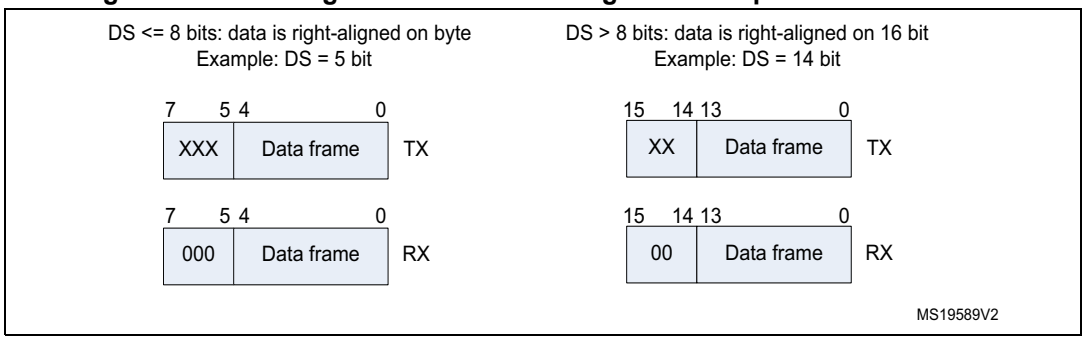


1. The order of data bits depends on LSBFIRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. The data frame size is chosen by using the DS bits. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception. Whatever the selected data frame size, read access to the FIFO must be aligned with the FRXTH level. When the SPIx_DR register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word (see *Figure 169*). During communication, only bits within the data frame are clocked and transferred.

Figure 169. Data alignment when data length is not equal to 8-bit or 16-bit



Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 8-bit data frame size.

24.5.6 SPI configuration

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated chapters. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CR1 register:
   a) Configure the serial clock baud rate using the BR[2:0] bits (Note *4*).
   b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (*2* 2).
   c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE can't be set at the same time).
   d) Configure the LSBFIRST bit to define the frame format (*2* 2).
   e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
   f) Configure SSM and SSI (*2* 2 and Note *3*).
   g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI_CR2 register:
   a) Configure the DS[3:0] bits to select the data length for the transfer.
   b) Configure SSOE (*2* 1 & 2 & 3).
   c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
   d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
   e) Configure the FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx_DR register.
   f) Initialize LDMA_TX and LDMA_RX bits if DMA is used in packed mode.
4. Write to SPI_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

Note: 1 Step is not required in slave mode.

2 Step is not required in TI mode.

3 Step is not required in NSSP mode.

4 The step is not required in slave mode except slave working at TI mode

24.5.7 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated chapter.

24.5.8 Data transmission and reception procedures

RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see *Section 24.5.13: CRC calculation*).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see *Section 24.5.12: TI mode*).

A read access to the SPIx_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, RXFIFO can store up to four data frames, whereas TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See *Figure 171* through *Figure 174*.

Another way to manage the data exchange is to use DMA (see *Section 10: DMA controller (DMA)*).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at *Section 24.5.10: SPI error flags*). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see *Section 24.5.4: Slave select (NSS) pin management*).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to prevent some dummy byte exchange (refer to the *Data packing* section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When the SPI is

disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional "dummy" data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC_APBiRSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions' streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

Note: If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.

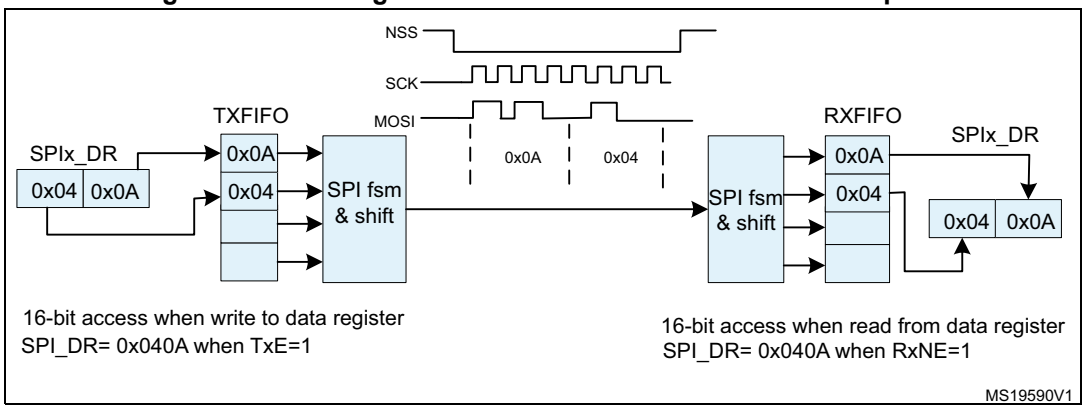
Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. *Figure 170* provides an example of data packing mode sequence handling. Two data frames are sent after the single 16-bit access the SPIx_DR register of the transmitter. This sequence can generate just one RXNE event in the receiver if the RXFIFO threshold is set to 16 bits (FRXTH=0). The receiver then has to access both data frames by a single 16-bit read of SPIx_DR as a response to this single RXNE event. The

RxFIFO threshold setting and the following read access must be always kept aligned at the receiver side, as data can be lost if it is not in line.

A specific problem appears if an odd number of such “fit into one byte” data frames must be handled. On the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPIx_DR is enough. The receiver has to change the Rx_FIFO threshold level for the last data frame received in the odd sequence of frames in order to generate the RXNE event.

Figure 170. Packing data in FIFO for transmission and reception



Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx_DR register.

See *Figure 171* through *Figure 174*.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Deepstop mode. The software must first wait until FTLVL[1:0]=00 and then until BSY=0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CR2 register, if DMA Tx and/or DMA Rx are used.

Packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPIx_CR2 register) packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel. If the DMA channel PSIZE value is equal to 16-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPIx_DR register.

If data packing mode is used and the number of data to transfer is not a multiple of two, the LDMA_TX/LDMA_RX bits must be set. The SPI then considers only one data for the transmission or reception to serve the last DMA transfer (for more details refer to *Data packing*.)

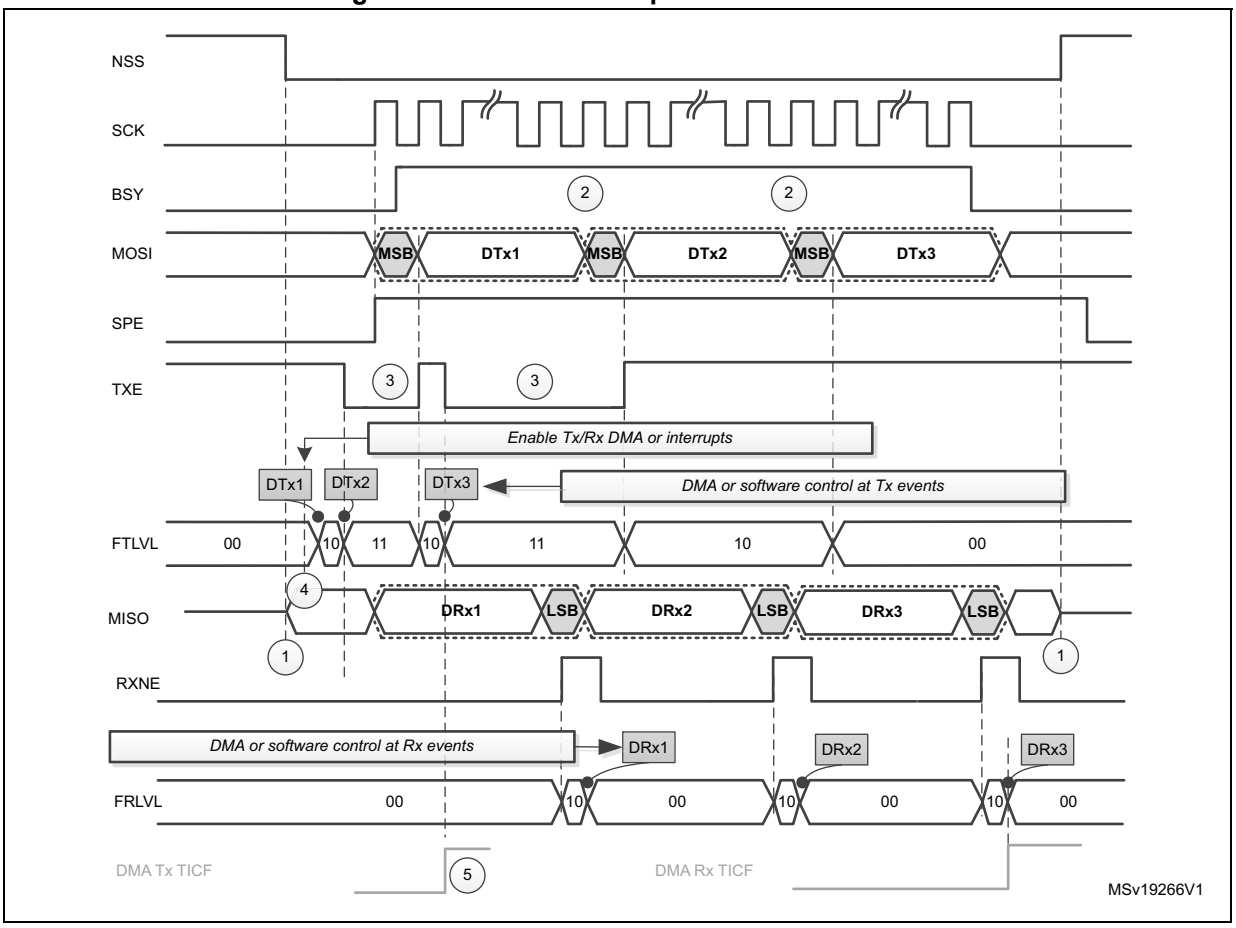
Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by pulling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for *Figure 171 on page 658* through *Figure 174 on page 661*.

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.
   At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx_TxCRCR and SPIx_RxCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).
   While the CRC value calculated in SPIx_TxCRCR is simply sent out by transmitter, received CRC information is loaded into RxFIFO and then compared with the SPIx_RxCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of RxFIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the TxFIFO is ¾ full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the TxFIFO becomes ½ full. This frame is stored into TxFIFO withan 8-bit access either by software or automatically by DMA when LDMA_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA_RX is set.

Figure 171. Master full duplex communication



Assumptions for master full duplex communication example:

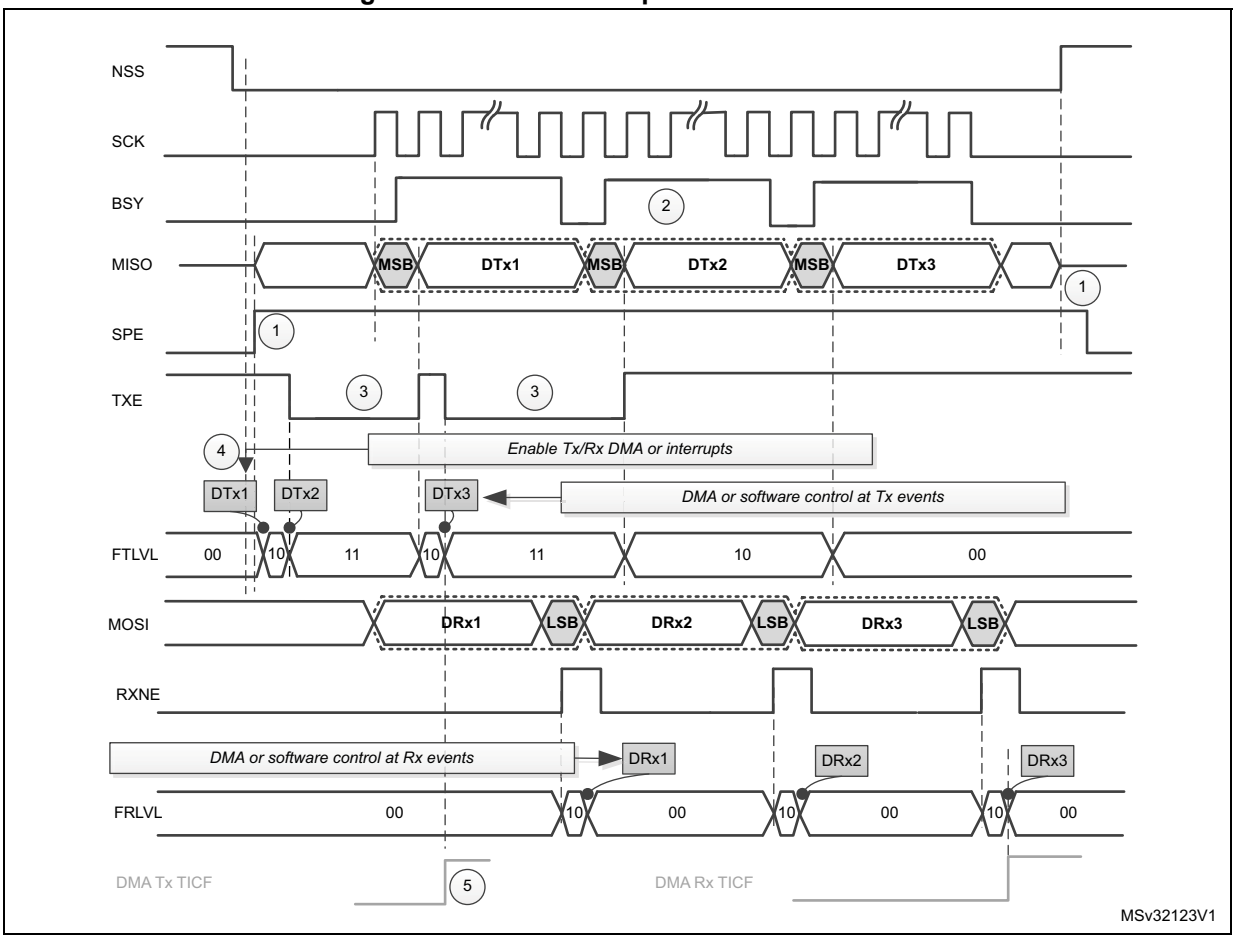
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *: Communication diagrams on page 657* for details about common assumptions and notes.

Figure 172. Slave full duplex communication



Assumptions for slave full duplex communication example:

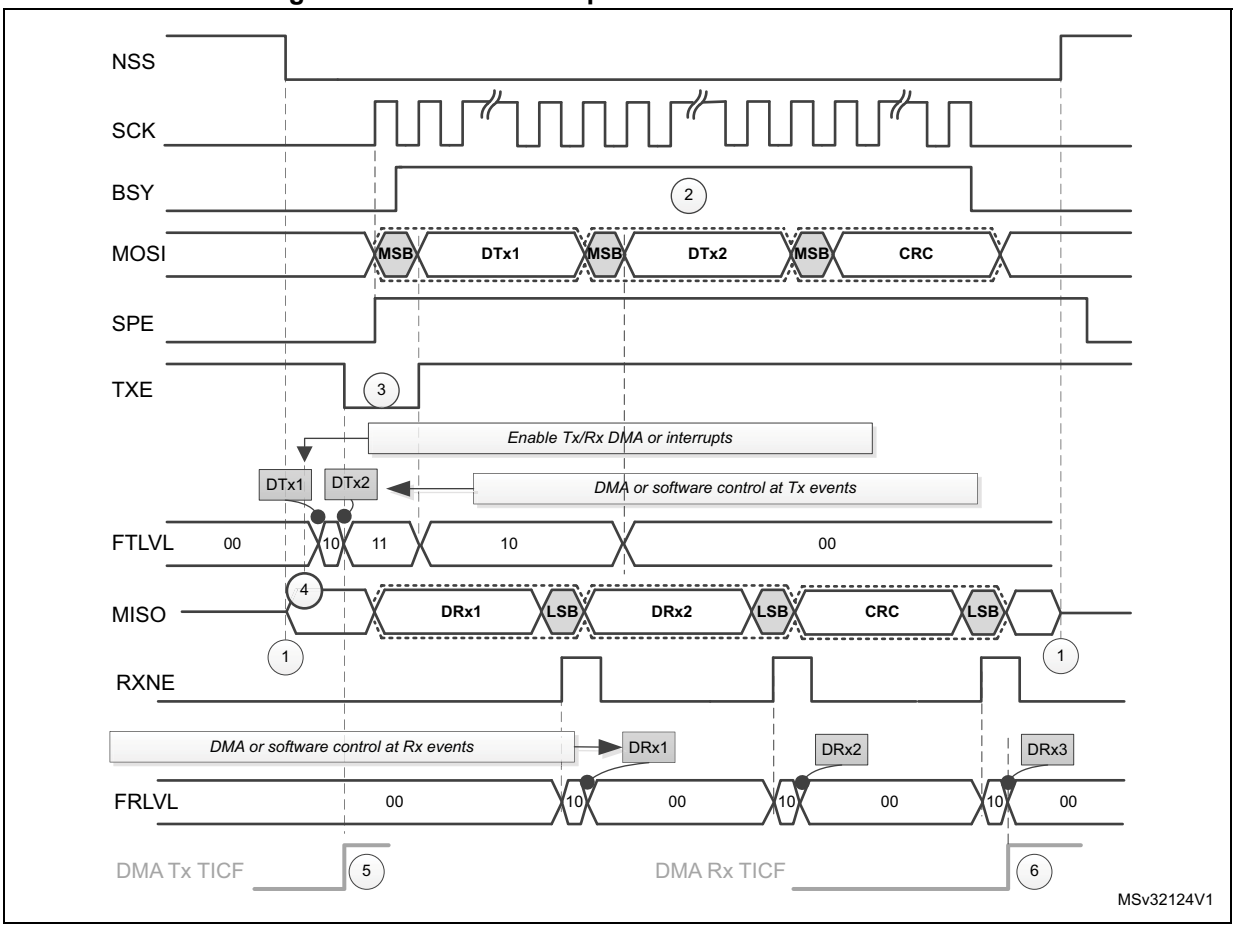
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *: Communication diagrams on page 657* for details about common assumptions and notes.

Figure 173. Master full duplex communication with CRC



Assumptions for master full duplex communication with CRC example:

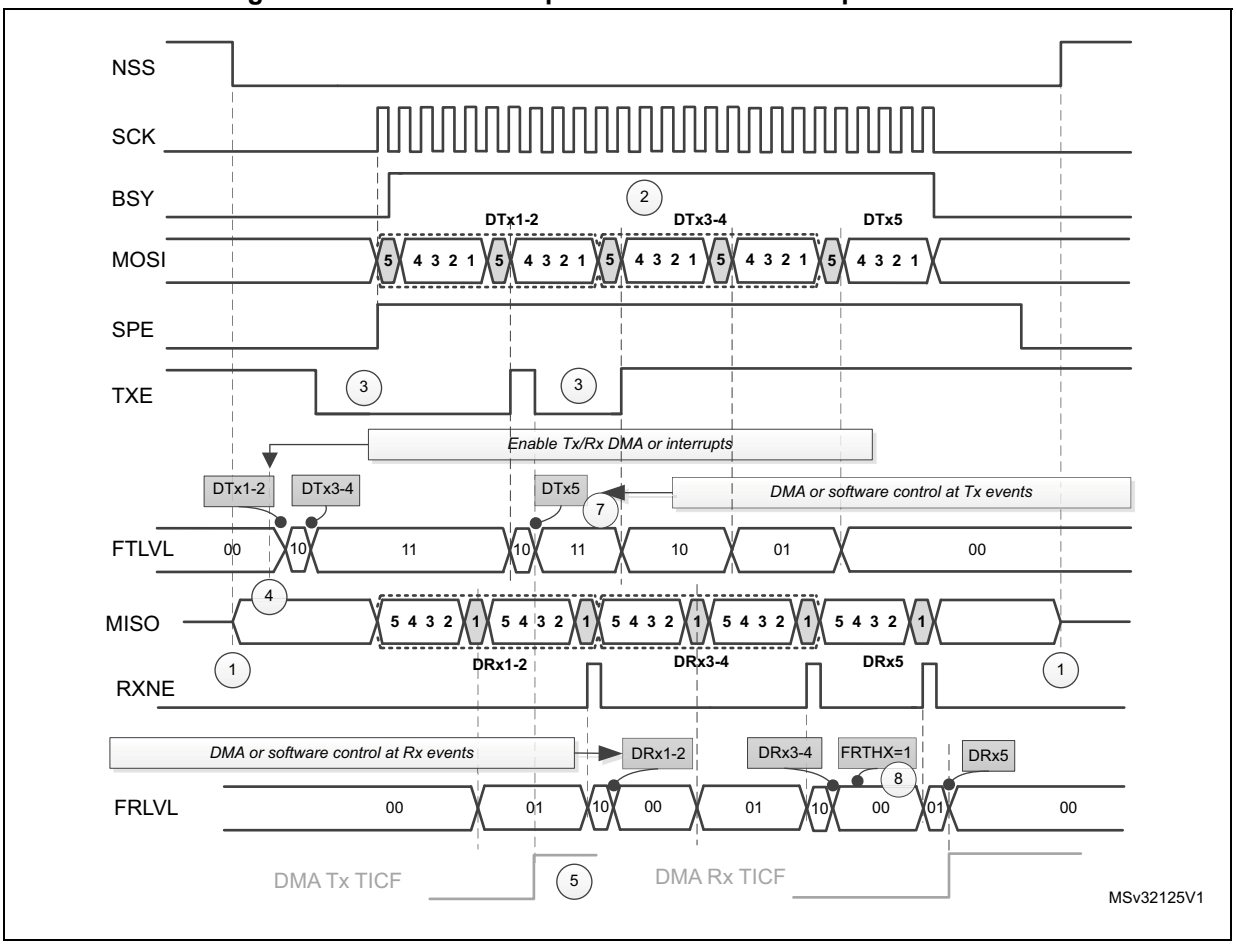
- Data size = 16 bit
- CRC enabled

If DMA is used:

- Number of Tx frames transacted by DMA is set to 2
- Number of Rx frames transacted by DMA is set to 3

See also *: Communication diagrams on page 657* for details about common assumptions and notes.

Figure 174. Master full duplex communication in packed mode



Assumptions for master full duplex communication in packed mode example:

- Data size = 5 bit
- Read/write FIFO is performed mostly by 16-bit access
- FRXTH=0

If DMA is used:

- Number of Tx frames to be transacted by DMA is set to 3
- Number of Rx frames to be transacted by DMA is set to 3
- PSIZE for both Tx and Rx DMA channel is set to 16-bit
- LDMA_TX=1 and LDMA_RX=1

See also *: Communication diagrams on page 657* for details about common assumptions and notes.

24.5.9 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note: When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

24.5.10 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see *Section 24.5.13: CRC calculation*).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI_DR register followed by a read access to the SPI_SR register.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx_SR register while the MODF bit is set.
2. Then write to the SPIx_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CR1 register is set. The CRCERR flag in the SPIx_SR register is set if the value received in the shift register does not match the receiver SPIx_RXCRCR value. The flag is cleared by the software.

TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

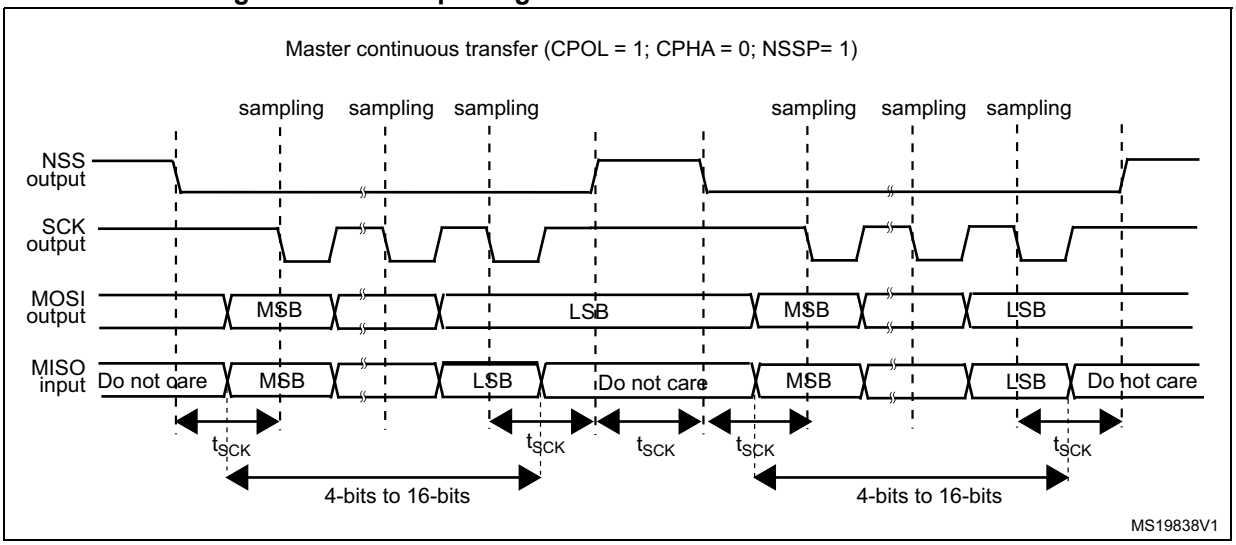
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

24.5.11 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

Figure 175 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 175. NSSP pulse generation in Motorola SPI master mode



Note: Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the *rising* edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

24.5.12 TI mode

TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx_CR1 and SPIx_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see *Figure 176*). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ ($t_{release}$) depends on internal resynchronization and on the

baud rate value set in through the BR[2:0] bits in the SPIx_CR1 register. It is given by the formula:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

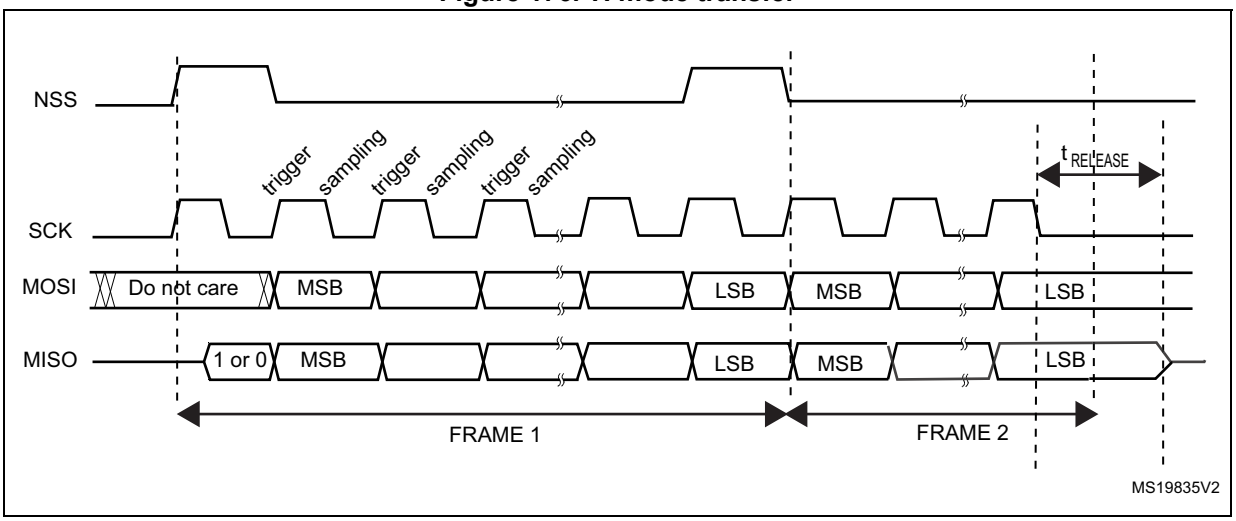
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bits or 5-bits, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

Figure 176: TI mode transfer shows the SPI communication waveforms when TI mode is selected.

Figure 176. TI mode transfer



24.5.13 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

Note: The polynomial value should only be odd. No even values are supported.

CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx_DR register. Then CRCNEXT bit has to be set in the SPIx_CR1 register to indicate that the CRC frame transaction follows after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPIx_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx_DR register in order to clear the RXNE flag.

CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

DMA_RX = Numb_of_data + 2

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA_RX bit needs managing if the number of data is odd.

Resetting the SPIx_TXCRC and SPIx_RXCRC values

The SPIx_TXCRC and SPIx_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

Note: When the SPI is in slave mode, the CRC calculator is sensitive to the SCK slave input clock as soon as the CRCEN bit is set, and this is the case whatever the value of the SPE bit. In order to avoid any wrong CRC calculation, the software must enable CRC calculation only when the clock is stable (in steady state). When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low between the data phase and the CRC phase.

24.6 SPI interrupts

During SPI communication an interrupts can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error

Interrupts can be enabled and disabled separately.

Table 116. SPI interrupt requests

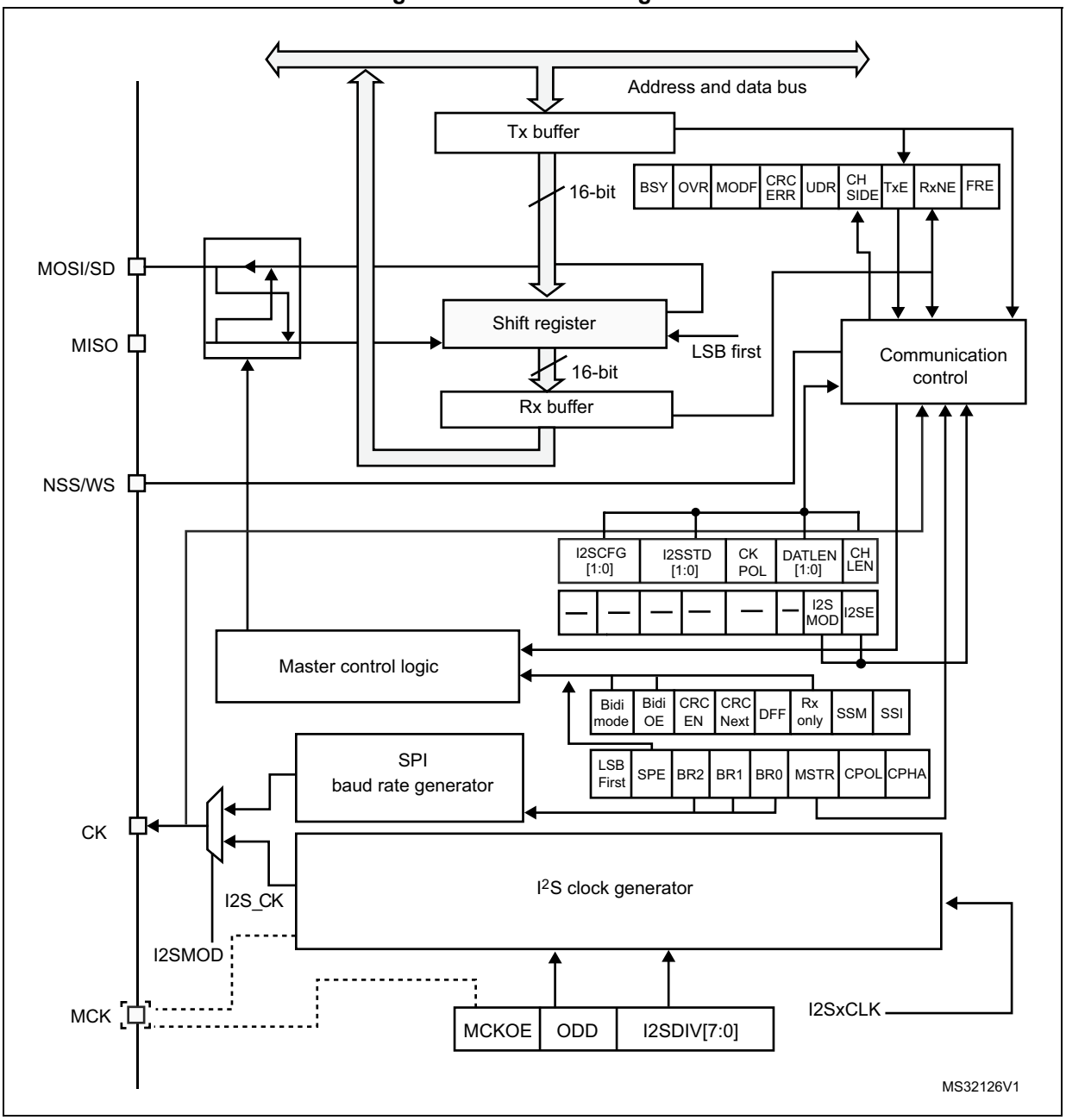
Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	

24.7 I²S functional description

24.7.1 I²S general description

The block diagram of the I²S is shown in *Figure 177*.

Figure 177. I²S block diagram



The SPI can function as an audio I²S interface when the I²S capability is enabled (by setting the I2SMOD bit in the SPIx_I2SCFGR register). This interface mainly uses the same pins, flags and interrupts as the SPI.

The I^2S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I^2S is configured in master mode (and when the MCKOE bit in the SPIx_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to 256 × f_S, where f_S is the audio sampling frequency.

The I^2S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I^2S mode. One is linked to the clock generator configuration SPIx_I2SPR and the other one is a generic I^2S configuration register SPIx_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx_CR1 register and all CRC registers are not used in the I^2S mode. Likewise, the SSOE bit in the SPIx_CR2 register and the MODF and CRCERR bits in the SPIx_SR are not used.

The I^2S uses the same SPI register for data transfer (SPIx_DR) in 16-bit wide mode.

24.7.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for transmission or reception. So, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPIx_SR register. Channel left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPIx_DR register or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 nonsignificant bits are extended to 32 bits with 0-bits (by hardware).

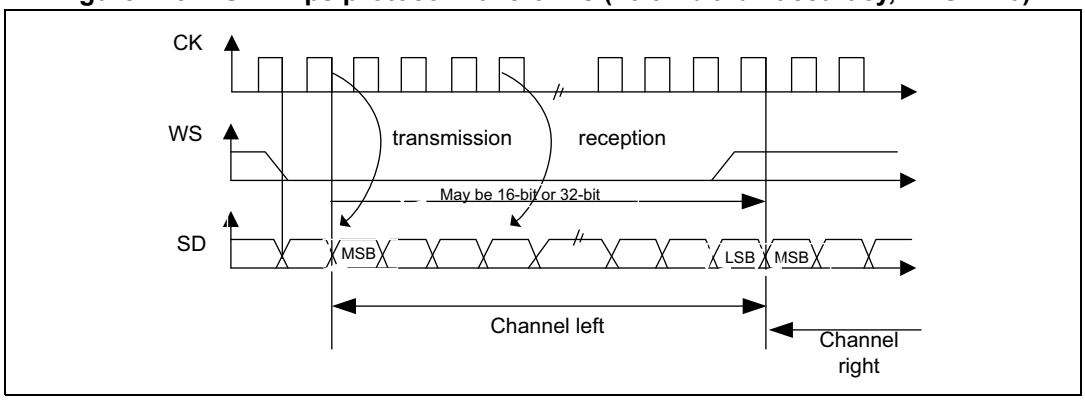
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I^{2}S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

I^{2}S Philips standard

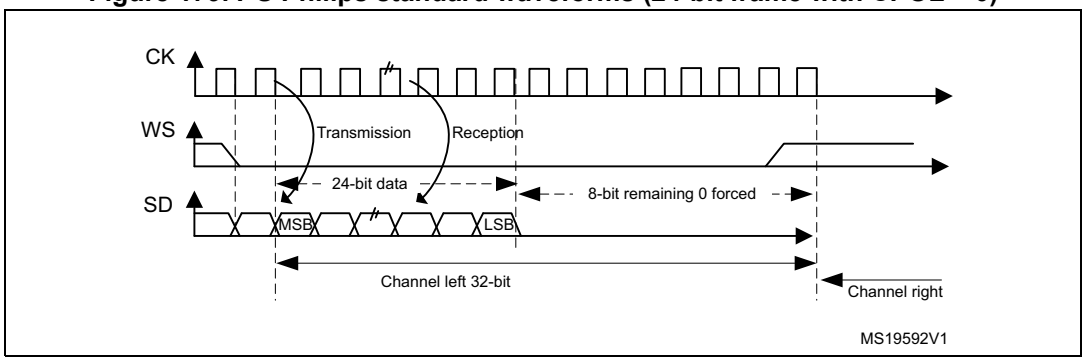
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Figure 178. I^{2}S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0)



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

Figure 179. I^{2}S Philips standard waveforms (24-bit frame with CPOL = 0)

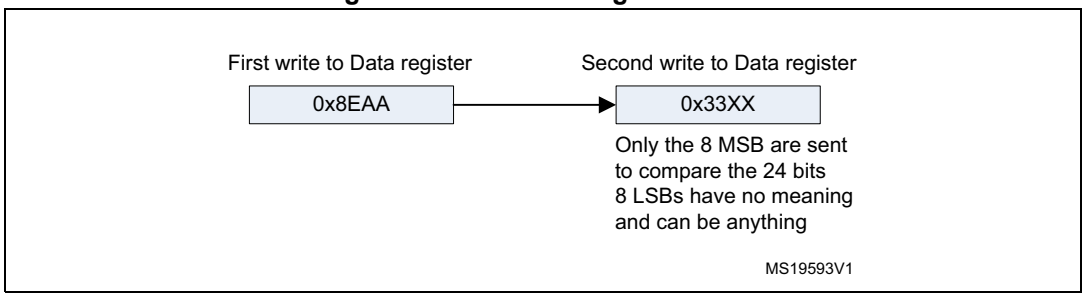


This mode needs two write or read operations to/from the SPIx_DR register.

- In transmission mode:

  If 0x8EAA33 has to be sent (24-bit):

Figure 180. Transmitting 0x8EAA33



- In reception mode:

  If data 0x8EAA33 is received:

Figure 181. Receiving 0x8EAA33

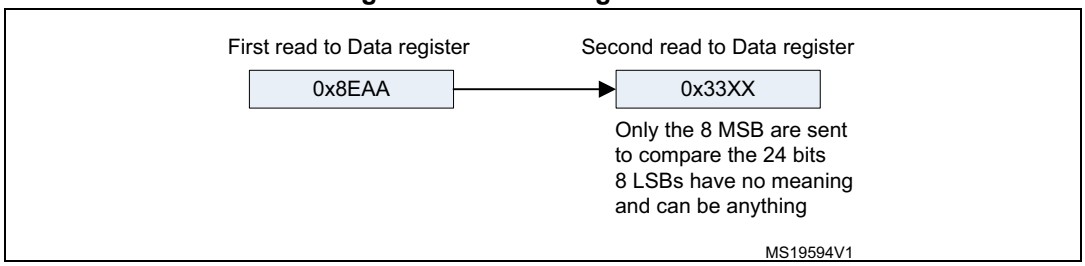
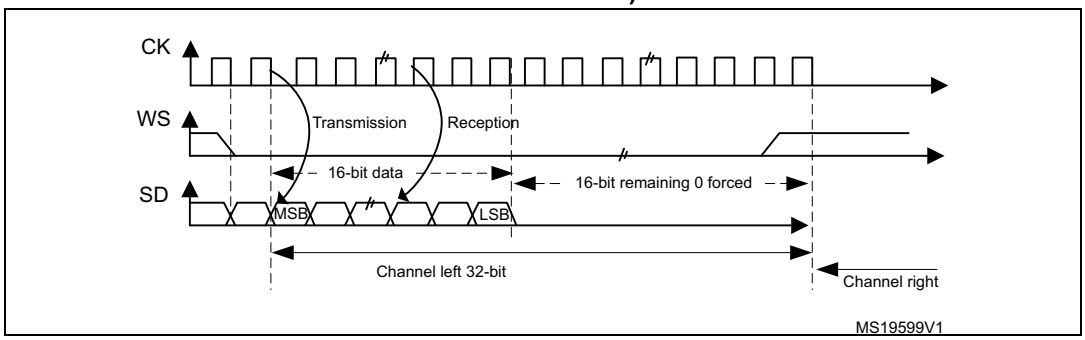


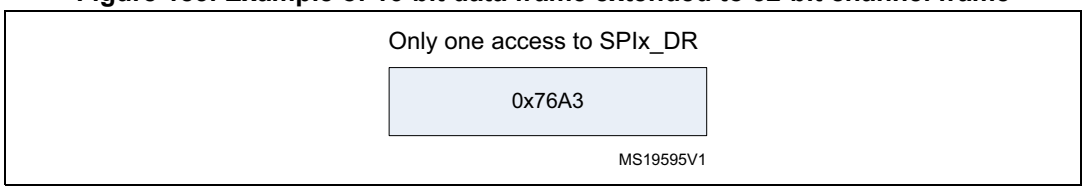
Figure 182. I²S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0)



When 16-bit data frame extended to 32-bit channel frame is selected during the I²S configuration phase, only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in *Figure 183* is required.

Figure 183. Example of 16-bit data frame extended to 32-bit channel frame



For transmission, each time an MSB is written to SPIx_DR, the TXE flag is set and its interrupt, if allowed, is generated to load the SPIx_DR register with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

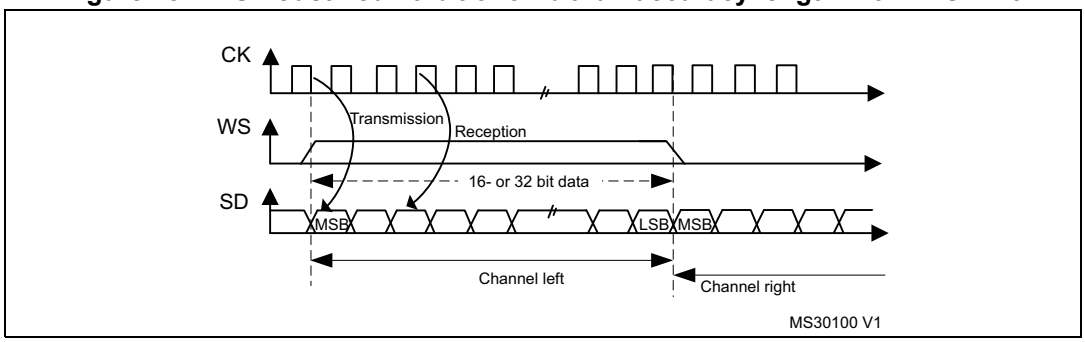
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

MSB justified standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

Figure 184. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

Figure 185. MSB justified 24-bit frame length with CPOL = 0

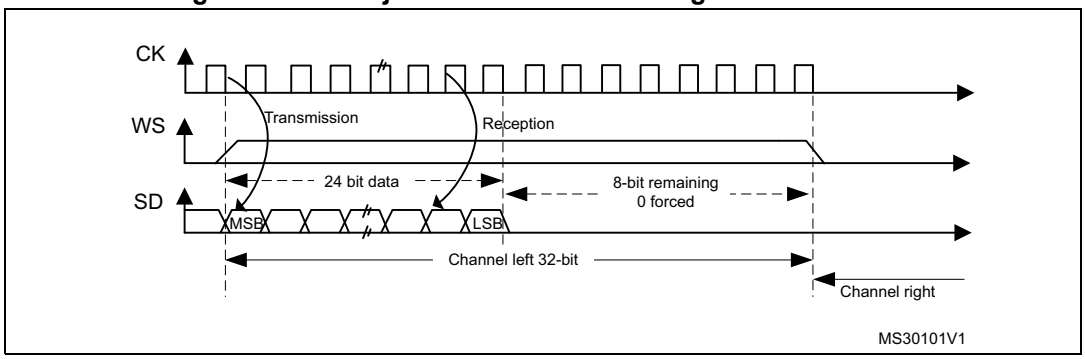
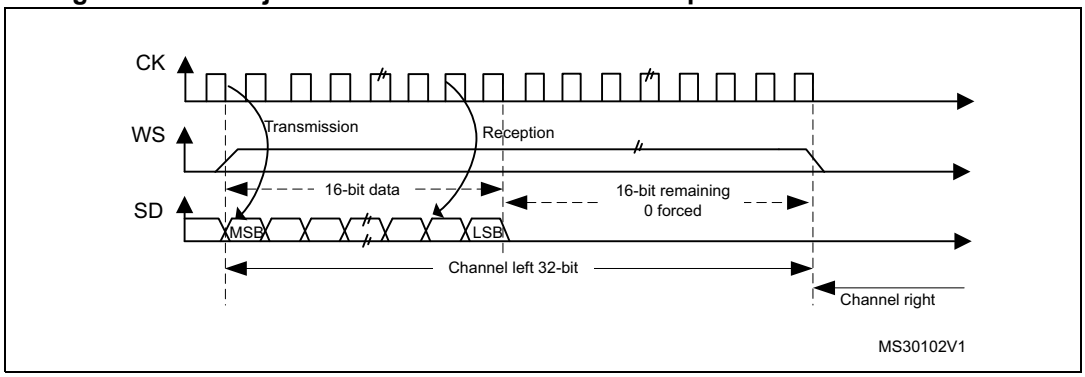


Figure 186. MSB justified 16-bit extended to 32-bit packet frame with CPOL = 0



LSB justified standard

This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

Figure 187. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0

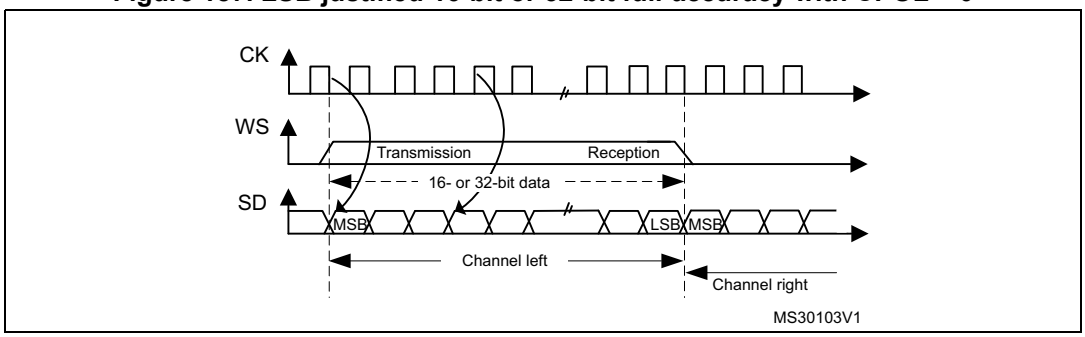
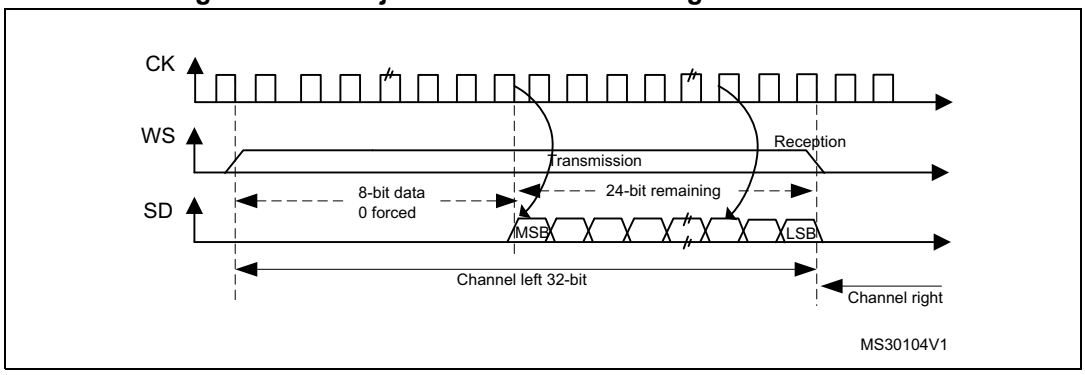


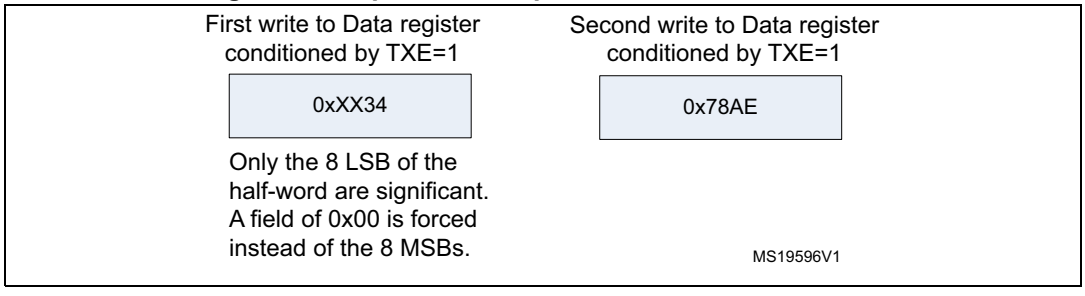
Figure 188. LSB justified 24-bit frame length with CPOL = 0



- In transmission mode:

  If data 0x3478AE have to be transmitted, two write operations to the SPIx_DR register are required by software or by DMA. The operations are shown below.

Figure 189. Operations required to transmit 0x3478AE



- In reception mode:

  If data 0x3478AE are received, two successive read operations from the SPIx_DR register are required on each RXNE event.

Figure 190. Operations required to receive 0x3478AE

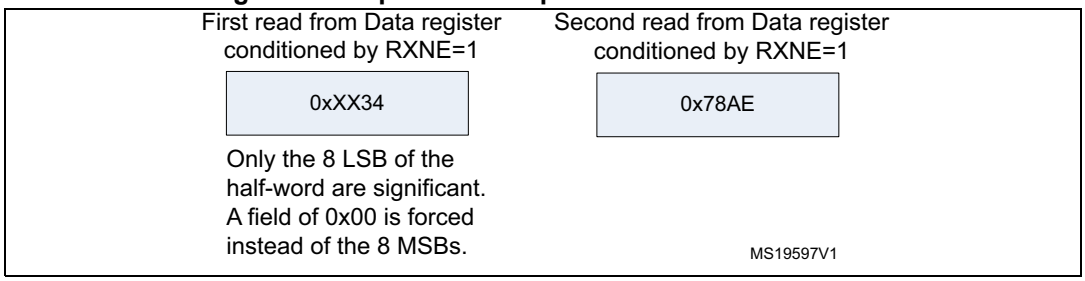
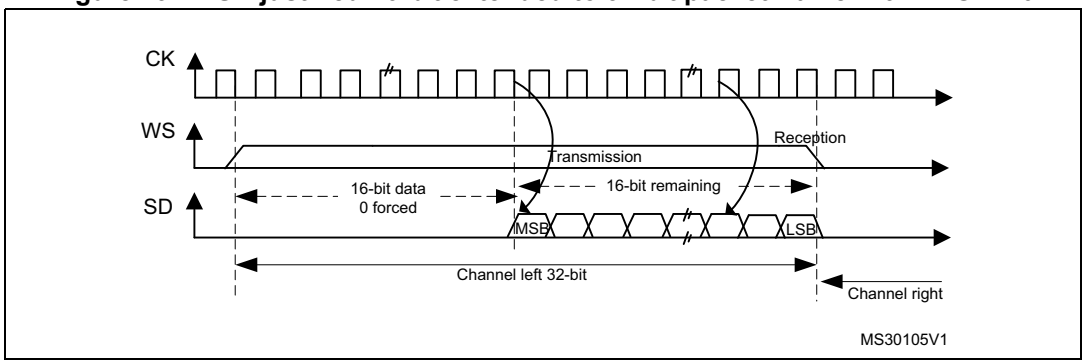


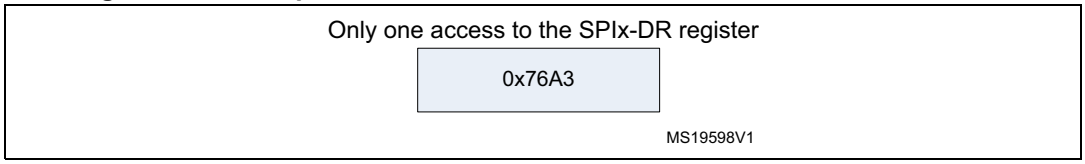
Figure 191. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0



When 16-bit data frame extended to 32-bit channel frame is selected during the I^2S configuration phase, Only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in *Figure 192* is required.

Figure 192. Example of 16-bit data frame extended to 32-bit channel frame



In transmission mode, when a TXE event occurs, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). The TXE flag is set again as soon as the effective data (0x76A3) is sent on SD.

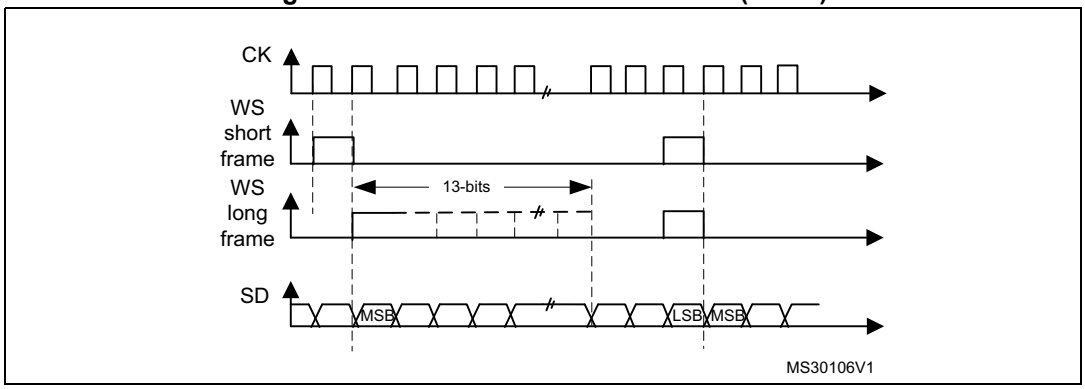
In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPIx_I2SCFGR register.

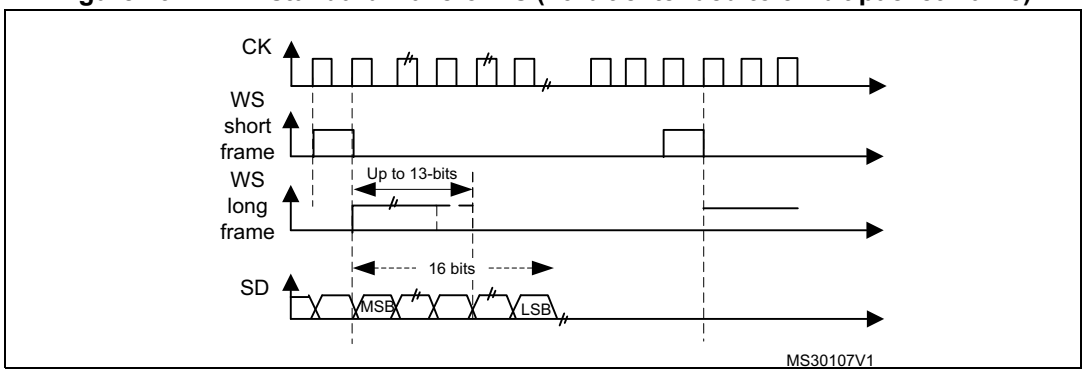
Figure 193. PCM standard waveforms (16-bit)



For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.

Figure 194. PCM standard waveforms (16-bit extended to 32-bit packet frame)



Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx_I2SCFGR register) even in slave mode.

24.7.3 Clock generator

The I^2S bitrate determines the dataflow on the I^2S data line and the I^2S clock signal frequency.

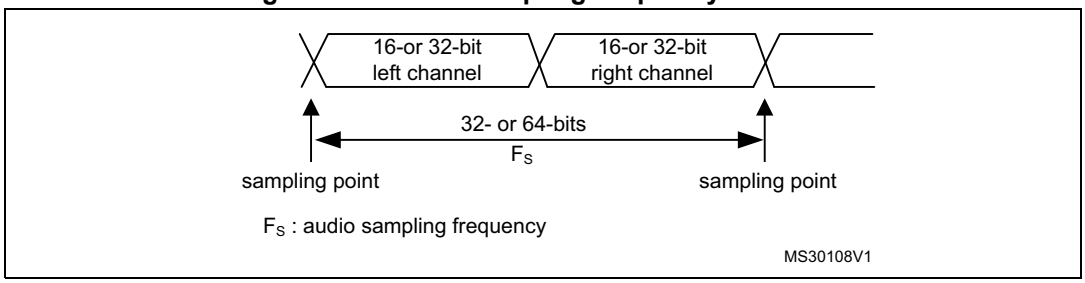
I^2S bitrate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I^2S bitrate is calculated as follows:

I^2S bitrate = 16 × 2 × f_S

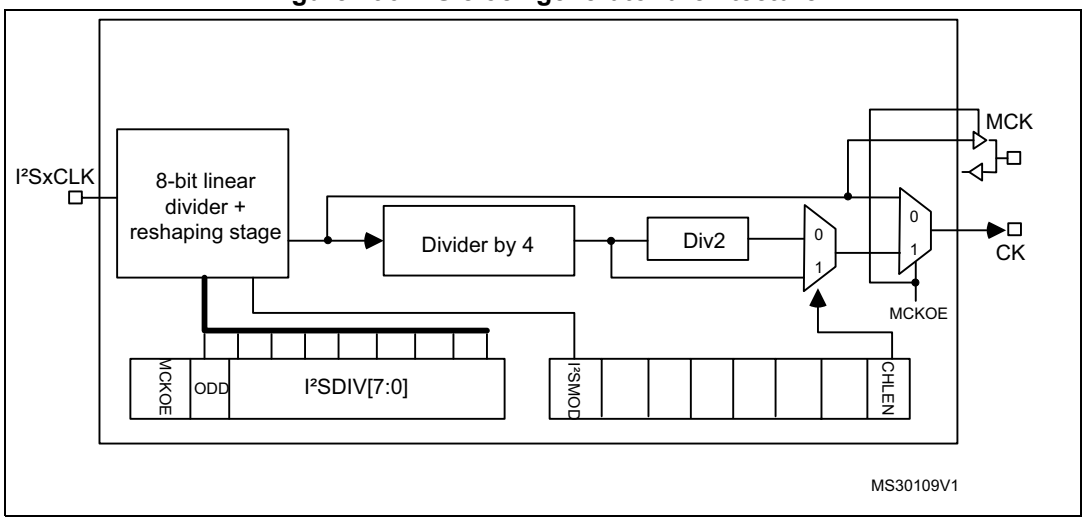
It is: I^2S bitrate = 32 x 2 x f_S if the packet length is 32-bit wide.

Figure 195. Audio sampling frequency definition



When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

Figure 196. I²S clock generator architecture



1. Where x can be 2 or 3.

Figure 196 presents the communication clock architecture. The I2Sx clock is always a 32 MHz frequency clock.

Warning: In addition, it is mandatory to keep I2SxCLK frequency higher or equal to the APB clock used by the SPI/I2S block. If this condition is not respected, SPI/I2S does not work.

The audio sampling frequency may be 192 kHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range). In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

f_S = I2SxCLK / [256 * ((2*I2SDIV)+ODD)] whatever the channel frame width (16-bit wide or 32-bit wide).

When the master clock is disabled (MCKOE bit cleared):

f_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD))] when the channel frame is 16-bit wide

f_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD))] when the channel frame is 32-bit wide

Table 117 provides example precision values for different clock configurations.

Note: Other configurations are possible that allow optimum clock precision.

Table 117. Audio frequency precision using I2SCLK = 64 MHz (continued)

I2SCLK (MHz)	Data length	I2SDIV	I2SODD	MCK	Target fs (Hz)	Real fs (KHz)	Error
64	16	5	0	No	192000	200000	4.1667%
64	32	2	1	No	192000	200000	4.1667%
64	16	10	1	No	96000	95238.1	0.7936%
64	32	5	0	No	96000	100000	4.167%
64	16	22	1	No	44100	44444.44	0.7811%
64	32	11	1	No	44100	43478.26	1.4098%
64	16	31	1	No	32000	31746.03	0.7937%
64	32	15	1	No	32000	32258.06	0.8065%
64	16	45	1	No	22050	21978.02	0.3264%
64	32	22	1	No	22050	22222.22	0.7811%
64	16	62	1	No	16000	160000	0%
64	32	31	1	No	16000	15873.032	0.7937%
64	16	90	1	No	11025	11049.72	0.2243%
64	32	45	1	No	11025	10989.01	0.3264%
64	16	125	0	No	8000	8000	0%
64	32	62	1	No	8000	8000	0%
64	16	2	1	Yes	48000	50000	4.1667%
64	32	2	1	Yes	48000	50000	4.1667%
64	16	3	0	Yes	44100	41666.667	5.5178%
64	32	3	0	Yes	44100	41666.667	5.5178%
64	16	4	0	Yes	32000	31250	2.3438%
64	32	4	0	Yes	32000	31250	2.3438%
64	16	5	1	Yes	22050	22727.27	3.0175%
64	32	5	1	Yes	22050	22727.27	3.0175%
64	16	8	0	Yes	16000	15625	2.3438%
64	32	8	0	Yes	16000	15625	2.3438%
64	16	11	1	Yes	11025	10869.57	1.4098%
64	32	11	1	Yes	11025	10869.57	1.4098%
64	16	15	1	Yes	8000	8064.516	0.8065%
64	32	15	1	Yes	8000	8064.516	0.8065%

Table 118. Audio frequency precision using I2SCLK = 32 MHz

I2SCLK (MHz)	Data length	I2SDIV	I2SODD	MCK	Target fs (Hz)	Real fs (KHz)	Error
32	16	5	0	No	96000	100000	4.1667%
32	32	2	1	No	96000	100000	4.1667%
32	16	10	1	No	48000	47619.0476	0.7936%
32	32	5	0	No	48000	50000	4.167%
32	16	11	1	No	44100	43478.261	1.410%
32	32	8	1	No	44100	45454.545	3.0715%
32	16	15	1	No	32000	32258.0645	0.806%
32	32	8	0	No	32000	31250	2.344%
32	16	22	1	No	22050	22222.22	0.781%
32	32	11	1	No	22050	21739.1304	1.410%
32	16	31	1	No	16000	15873.0159	0.794%
32	32	15	1	No	16000	16129.032	0.806%
32	16	45	1	No	11025	10989.011	0.326%
32	32	22	1	No	11025	11111.111	0.781%
32	16	62	1	No	8000	8000	0%
32	32	47	0	No	8000	7936.508	0.794%
32	16	1	1	Yes	48000	41666.667	13.194%
32	32	1	1	Yes	48000	41666.667	13.194%
32	16	1	1	Yes	44100	41666.667	5.518%
32	32	1	1	Yes	44100	41666.667	5.518%
32	16	2	0	Yes	32000	31250	2.3438%
32	32	2	0	Yes	32000	31250	2.3438%
32	16	3	0	Yes	22050	20833.333	5.5178%
32	32	3	0	Yes	22050	20833.333	5.5178%
32	16	4	0	Yes	16000	15625	2.3438%
32	32	4	0	Yes	16000	15625	2.3438%
32	16	5	1	Yes	11025	11363.6364	3.0715%
32	32	5	1	Yes	11025	11363.6364	3.0715%
32	16	8	0	Yes	8000	7812.5	2.344%
32	32	8	0	Yes	8000	7812.5	2.344%

Table 119. Audio frequency precision using I2SCLK = 16 MHz

I2SCLK (MHz)	Data length	I2SDIV	I2SODD	MCK	Target fs (Hz)	Real fs (KHz)	Error
16	16	2	1	No	96000	100000	4.1667%
16	32	2	0	No	96000	62500	34.890%
16	16	4	1	No	48000	50000	4.167%
16	32	2	1	No	48000	50000	4.167%
16	16	5	1	No	44100	45454.545	3.0715%
16	32	3	0	No	44100	41666.67	5.518%
16	16	8	0	No	32000	31250	2.344%
16	32	4	0	No	32000	31250	2.344%
16	16	11	1	No	22050	21739.13	1.410%
16	32	5	1	No	22050	22727.27	3.071%
16	16	15	1	No	16000	16129.032	0.806%
16	32	15	1	No	16000	15625	2.344%
16	16	22	1	No	11025	11111.111	0.781%
16	32	11	1	No	11025	10869.57	1.409%
16	16	31	1	No	8000	7936.51	0.794%
16	32	15	1	No	8000	8064.52	0.806%
16	16	N/A	N/A	Yes	48000	N/A	-
16	32	N/A	N/A	Yes	48000	N/A	-
16	16	N/A	N/A	Yes	44100	N/A	-
16	32	N/A	N/A	Yes	44100	N/A	-
16	16	N/A	N/A	Yes	32000	N/A	-
16	32	N/A	N/A	Yes	32000	N/A	-
16	16	3	0	Yes	22050	N/A	-
16	32	3	0	Yes	22050	N/A	-
16	16	2	0	Yes	16000	15625	2.3438%
16	32	2	0	Yes	16000	15625	2.3438%
16	16	3	0	Yes	11025	10416.67	5.518%
16	32	3	0	Yes	11025	10416.67	5.518%
16	16	4	0	Yes	8000	7812.5	2.344%
16	32	4	0	Yes	8000	7812.5	2.344%

24.7.4 I²S master mode

The I²S can be configured in master mode. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, controlled by the MCKOE bit in the SPIx_I2SPR register.

Procedure

1. Select the I2SDIV[7:0] bits in the SPIx_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPIx_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPIx_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to *Section 24.7.3: Clock generator*).
3. Set the I2SMOD bit in the SPIx_I2SCFGR register to activate the I²S functions and choose the I²S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I²S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.
4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
5. The I2SE bit in SPIx_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPIx_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Lets assume the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the right channel have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

For more details about the write operations depending on the I²S standard mode selected, refer to *Section 24.7.2: Supported audio protocols*).

To ensure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission.

To switch off the I²S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in *Section 24.7.4: I²S master mode*), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPIx_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I²S cell.

For more details about the read operations depending on the I²S standard mode selected, refer to *Section 24.7.2: Supported audio protocols*.

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I²S, specific actions are required to ensure that the I²S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait 17 I²S clock cycles (using a software loop)
  c) Disable the I²S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I²S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
  a) Wait for the last RXNE
  b) Then wait 1 I²S clock cycle (using a software loop)
  c) Disable the I²S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I²S:
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait one I²S clock cycle (using a software loop)
  c) Disable the I²S (I2SE = 0)

Note: The BSY flag is kept low during transfers.

24.7.5 I²S slave mode

For the slave configuration, the I²S can be configured in transmission or reception mode. The operating mode is following mainly the same rules as described for the I²S master configuration. In slave mode, there is no clock to be generated by the I²S interface. The

clock and WS signals are input from the external master connected to the I^{2}S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPIx_I2SCFGR register to select I^{2}S mode and choose the I^{2}S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPIx_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
3. The I2SE bit in SPIx_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I^{2}S data register has to be loaded before the master initiates the communication.

For the I^{2}S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I^{2}S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer.

For more details about the write operations depending on the I^{2}S standard mode selected, refer to *Section 24.7.2: Supported audio protocols*.

To secure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPIx_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPIx_CR2 register, an interrupt is generated when the UDR flag in the SPIx_SR register goes high. In this case, it is mandatory to switch off the I^{2}S and to restart a data transfer starting from the left channel.

To switch off the I^{2}S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in *Section 24.7.5: I²S slave mode*), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPIx_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPIx_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from the SPIx_DR register. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

For more details about the read operations depending the I²S standard mode selected, refer to *Section 24.7.2: Supported audio protocols*.

If data are received while the preceding received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I²S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

Note: The external master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

24.7.6 I²S error flags

There are three error flags for the I²S cell.

Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx_DR. It is available when the I2SMOD bit in the SPIx_I2SCFGR register is set. An interrupt may be generated if the ERRIE bit in the SPIx_CR2 register is set.
The UDR bit is cleared by a read operation on the SPIx_SR register.

Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from the SPIx_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx_DR register followed by a read access to the SPIx_SR register.

Frame error flag (FRE)

This flag can be set by hardware only if the I^2S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the I^2S slave device:

1. Disable the I^2S.
2. Enable it again when the correct level is detected on the WS line (WS line is high in I^2S mode or low for MSB- or LSB-justified or PCM modes.

Desynchronization between master and slave devices may be due to noisy environment on the SCK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

24.7.7 DMA features

In I^2S mode, the DMA works in exactly the same way as it does in SPI mode. There is no difference except that the CRC feature is not available in I^2S mode since there is no data transfer protection system.

24.8 I^2S interrupts

Table 120 provides the list of I^2S interrupts.

Table 120. I^2S interrupt requests

Interrupt event	Event flag	Enable control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overrun error	OVR	ERRIE
Underrun error	UDR	
Frame error flag	FRE	

24.9 SPI and I²S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition by can be accessed by 8-bit access.

24.9.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE:** Bidirectional data mode enable. This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

Note: This bit is not used in I²S mode.

Bit 14 **BIDIOE:** Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used. This bit is not used in I²S mode.

Bit 13 **CRCEN:** Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation Enabled

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation. This bit is not used in I²S mode.

Bit 12 **CRCNEXT:** Transmit CRC next

0: Next transmit value is from Tx buffer

1: Next transmit value is from Tx CRC register

Note: This bit has to be written as soon as the last data is written in the SPIx_DR register. This bit is not used in I²S mode.

Bit 11 **CRCL:** CRC length

This bit is set and cleared by software to select the CRC length.

0: 8-bit CRC length

1: 16-bit CRC length

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation. This bit is not used in I²S mode.

Bit 10 **RXONLY:** Receive only mode enabled.

This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active.This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

0: Full duplex (Transmit and receive)

1: Output disabled (Receive-only mode)

Note: This bit is not used in I^2S mode.

Bit 9 **SSM:** Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

0: Software slave management disabled

1: Software slave management enabled

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 8 **SSI:** Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 7 **LSBFIRST***:* Frame format

0: data is transmitted / received with the MSB first

1: data is transmitted / received with the LSB first

Note: 1. This bit should not be changed when communication is ongoing.

2. This bit is not used in I^2S mode and SPI TI mode.

Bit 6 **SPE:** SPI enable

0: Peripheral disabled

1: Peripheral enabled

Note: When disabling the SPI, follow the procedure described in Procedure for disabling the SPI on page 653.

This bit is not used in I^2S mode.

Bits 5:3 **BR[2:0]:** Baud rate control

000: $f_{PCLK}/2$

001: $f_{PCLK}/4$

010: $f_{PCLK}/8$

011: $f_{PCLK}/16$

100: $f_{PCLK}/32$

101: $f_{PCLK}/64$

110: $f_{PCLK}/128$

111: $f_{PCLK}/256$

Note: These bits should not be changed when communication is ongoing.

This bit is not used in I^2S mode.

Bit 2 **MSTR:** Master selection
0: Slave configuration
1: Master configuration
Note: This bit should not be changed when communication is ongoing.
This bit is not used in I^2S mode.

Bit1 **CPOL:** Clock polarity
0: CK to 0 when idle
1: CK to 1 when idle
Note: This bit should not be changed when communication is ongoing.
This bit is not used in I^2S mode and SPI TI mode.

Bit 0 **CPHA:** Clock phase
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge
Note: This bit should not be changed when communication is ongoing.
This bit is not used in I^2S mode and SPI TI mode.

24.9.2 SPI control register 2 (SPIx_CR2)

Address offset: 0x04

Reset value: 0x0700

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LDMA_TX	LDMA_RX	FRXTH	DS [3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **LDMA_TX:** Last DMA transfer for transmission
This bit is used in data packing mode, to define if the total number of data to transmit by DMA is odd or even. It has significance only if the TXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).
0: Number of data to transfer is even
1: Number of data to transfer is odd
Note: Refer to Procedure for disabling the SPI on page 653 if the CRCEN bit is set.
This bit is not used in I²S mode.

Bit 13 **LDMA_RX**: Last DMA transfer for reception
This bit is used in data packing mode, to define if the total number of data to receive by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).
0: Number of data to transfer is even
1: Number of data to transfer is odd
Note: Refer to Procedure for disabling the SPI on page 653 if the CRCEN bit is set.
This bit is not used in I²S mode.

Bit 12 **FRXTH**: FIFO reception threshold

FRXTH shall be set according the read access (16-bit or 8-bit) to the FIFO.

This bit is used to set the threshold of the RXFIFO that triggers an RXNE event

0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit)

1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)

Note: This bit is not used in I²S mode.

Bit 11:8 **DS [3:0]**: Data size

These bits configure the data length for SPI transfers:

0000: Not used
0001: Not used
0010: Not used
0011: 4-bit
0100: 5-bit
0101: 6-bit
0110: 7-bit
0111: 8-bit
1000: 9-bit
1001: 10-bit
1010: 11-bit
1011: 12-bit
1100: 13-bit
1101: 14-bit
1110: 15-bit
1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111”(8-bit).

Note: This bit is not used in I²S mode.

Bit 7 **TXEIE:** Tx buffer empty interrupt enable

0: TXE interrupt masked

1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable

0: RXNE interrupt masked

1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode and UDR, OVR, and FRE in I^2S mode).

0: Error interrupt is masked

1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

0: SPI Motorola mode

1 SPI TI mode

Note: This bit must be written only when the SPI is disabled (SPE=0).
This bit is not used in I^2S mode.

Bit 3 **NSSP**: NSS pulse management

This bit is used in master mode only. it allow the SPI to generate an NSS pulse between two consecutive data when doing continuous transfers. In the case of a single data transfer, it forces the NSS pin high level after the transfer.

It has no meaning if CPHA = ’1’, or FRF = ’1’.

0: No NSS pulse

1: NSS pulse generated

Note: 1. This bit must be written only when the SPI is disabled (SPE=0).

2. This bit is not used in I^2S mode and SPI TI mode.

Bit 2 **SSOE:** SS output enable

0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration

1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 1 **TXDMAEN:** Tx buffer DMA enable

When this bit is set, a DMA request is generated whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN:** Rx buffer DMA enable

When this bit is set, a DMA request is generated whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

24.9.3 SPI status register (SPIx_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTLVL[1:0]		FRLVL[2:0]		FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0	r	r	r	r

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:11 **FTLVL[1:0]:** FIFO Transmission Level

These bits are set and cleared by hardware.

00: FIFO empty
01: 1/4 FIFO
10: 1/2 FIFO
11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)

Note: These bits are not used in I²S mode.

Bits 10:9 **FRLVL[1:0]**: FIFO reception level

These bits are set and cleared by hardware.

00: FIFO empty
01: 1/4 FIFO
10: 1/2 FIFO
11: FIFO full

Note: These bits are not used in I²S mode and in SPI receive-only mode while CRC calculation is enabled.

Bits 8 **FRE**: Frame format error

This flag is used for SPI in TI slave mode and I^2S slave mode. Refer to *Section 24.5.10: SPI error flags* and *Section 24.7.6: I^2S error flags*.

This flag is set by hardware and reset when SPIx_SR is read by software.

0: No frame format error
1: A frame format error occurred

Bit 7 **BSY:** Busy flag

0: SPI (or I2S) not busy
1: SPI (or I2S) is busy in communication or Tx buffer is not empty

This flag is set and cleared by hardware.

Note: The BSY flag must be used with caution: refer to Section 24.5.9: SPI status flags and Procedure for disabling the SPI on page 653.

Bit 6 **OVR:** Overrun flag

0: No overrun occurred
1: Overrun occurred

This flag is set by hardware and reset by a software sequence. Refer to *I^2S error flags on page 683* for the software sequence.

Bit 5 **MODF:** Mode fault

0: No mode fault occurred
1: Mode fault occurred

This flag is set by hardware and reset by a software sequence. Refer to *Section : Mode fault (MODF)* for the software sequence.

Bit 4 **CRCERR:** CRC error flag
0: CRC value received matches the SPIx_RXCRCR value
1: CRC value received does not match the SPIx_RXCRCR value
This flag is set by hardware and cleared by software writing 0.

Bit 3 **UDR:** Underrun flag
0: No underrun occurred
1: Underrun occurred
This flag is set by hardware and reset by a software sequence. Refer to *I^2S error flags on page 683* for the software sequence.
Note: This bit is not used in SPI mode.

Bit 2 **CHSIDE**: Channel side
0: Channel Left has to be transmitted or has been received
1: Channel Right has to be transmitted or has been received
Note: This bit is not used in SPI mode. It has no significance in PCM mode.

Bit 1 **TXE:** Transmit buffer empty
0: No more empty space in Tx buffer. (software shall not write data to the Tx buffer).
1: At least one empty space in Tx buffer. (software may write data to the Tx buffer).

Bit 0 **RXNE:** Receive buffer not empty
0: Rx buffer empty
1: Rx buffer not empty

24.9.4 SPI data register (SPIx_DR)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DR[15:0]:** Data register

Data received or to be transmitted

The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses TxFIFO (See Section 24.5.8: Data transmission and reception procedures).

Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.

24.9.5 SPI CRC polynomial register (SPIx_CRCPR)

Address offset: 0x10

Reset value: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CRCPOLY[15:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0007h) is the reset value of this register. Another polynomial can be configured as required.

Note: The polynomial value should be odd only. No even value is supported.

24.9.6 SPI Rx CRC register (SPIx_RXCRCR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RxCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **RXCRC[15:0]:** Rx CRC register

When CRC calculation is enabled, the RxCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPIx_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

A read to this register when the BSY Flag is set could return an incorrect value.

24.9.7 SPI Tx CRC register (SPIx_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **TxCRC[15:0]:** Tx CRC register

When CRC calculation is enabled, the TxCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPIx_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY flag is set could return an incorrect value.

These bits are not used in I^2S mode.

24.9.8 SPIx_I²S configuration register (SPIx_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ASTRTEN	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
			rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved: Forced to 0 by hardware

Bit 12 **ASTRTEN**: Asynchronous start enable.

0: The Asynchronous start is disabled. When the I2S is enabled in slave mode, the I2S slave starts the transfer when the I2S clock is received and an appropriate transition (depending on the protocol selected) is detected on the WS signal.

1: The Asynchronous start is enabled. When the I2S is enabled in slave mode, the I2S slave starts immediately the transfer when the I2S clock is received from the master without checking the expected transition of WS signal.

Note: The appropriate transition is a falling edge on WS signal when I2S Philips Standard is used, or a rising edge for other standards.

Bit 11 **I2SMOD**: I2S mode selection

0: SPI mode is selected

1: I2S mode is selected

Note: This bit should be configured when the SPI is disabled.

Bit 10 **I2SE**: I2S enable

0: I²S peripheral is disabled

1: I²S peripheral is enabled

Note: This bit is not used in SPI mode.

Bits 9:8 **I2SCFG**: I2S configuration mode

00: Slave - transmit

01: Slave - receive

10: Master - transmit

11: Master - receive

Note: These bits should be configured when the I²S is disabled.
They are not used in SPI mode.

Bit 7 **PCMSYNC**: PCM frame synchronization

0: Short frame synchronization

1: Long frame synchronization

Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used).
It is not used in SPI mode.

Bit 6 Reserved: forced at 0 by hardware

Bits 5:4 **I2SSTD**: I^2S standard selection
00: I^2S Philips standard.
01: MSB justified standard (left justified)
10: LSB justified standard (right justified)
11: PCM standard
For more details on I^2S standards, refer to *Section 24.7.2 on page 669*
Note: For correct operation, these bits should be configured when the I^2S is disabled. They are not used in SPI mode.

Bit 3 **CKPOL**: Steady state clock polarity
0: I^2S clock steady state is low level
1: I^2S clock steady state is high level
Note: For correct operation, this bit should be configured when the I^2S is disabled. It is not used in SPI mode.

Bits 2:1 **DATLEN**: Data length to be transferred
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: For correct operation, these bits should be configured when the I^2S is disabled. They are not used in SPI mode.

Bit 0 **CHLEN**: Channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide
The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.
Note: For correct operation, this bit should be configured when the I^2S is disabled. It is not used in SPI mode.

24.9.9 SPIx_I²S prescaler register (SPIx_I2SPR)

Address offset: 0x20

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
						rw	rw	rw							

Bits 15:10 Reserved: Forced to 0 by hardware

Bit 9 **MCKOE**: Master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Note: This bit should be configured when the I²S is disabled. It is used only when the I²S is in master mode.

It is not used in SPI mode.

Bit 8 **ODD**: Odd factor for the prescaler

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2)+1

Refer to *Section 24.7.3 on page 675*

Note: This bit should be configured when the I²S is disabled. It is used only when the I²S is in master mode.

It is not used in SPI mode.

Bits 7:0 **I2SDIV**: I²S linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to *Section 24.7.3 on page 675*

Note: These bits should be configured when the I²S is disabled. They are used only when the I²S is in master mode.

They are not used in SPI mode.

24.9.10 SPI/I2S register map

Table 121 shows the SPI/I2S register map and reset values.

Table 121. SPI register map and reset values

Offset	Register	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
0x00	**SPIx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPIx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	*SSOE*	TXDMAEN	RXDMAEN
	Reset value																		0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0x08	**SPIx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	1	0
0x0C	**SPIx_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPIx_CRCPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPIx_RXCRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPIx_TXCRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**SPIx_I2SCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASTRTEN	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset value																				0	0	0	0	0	0		0	0	0	0	0	0
0x20	**SPIx_I2SPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

25 Radio IP

25.1 Overview

This chapter describes the Radio IP embedded in the STM32WB09xE product.

The Radio controller MR_BLE manages the Bluetooth® LE protocol with HW add-on to support some new Bluetooth specification features, and controls the RF analog module.

25.1.1 Architecture overview of the Radio controller MR_BLE IP

As shown in *Figure 197: Radio controller MR_BLE architecture overview*, the Radio controller MR_BLE IP contains:

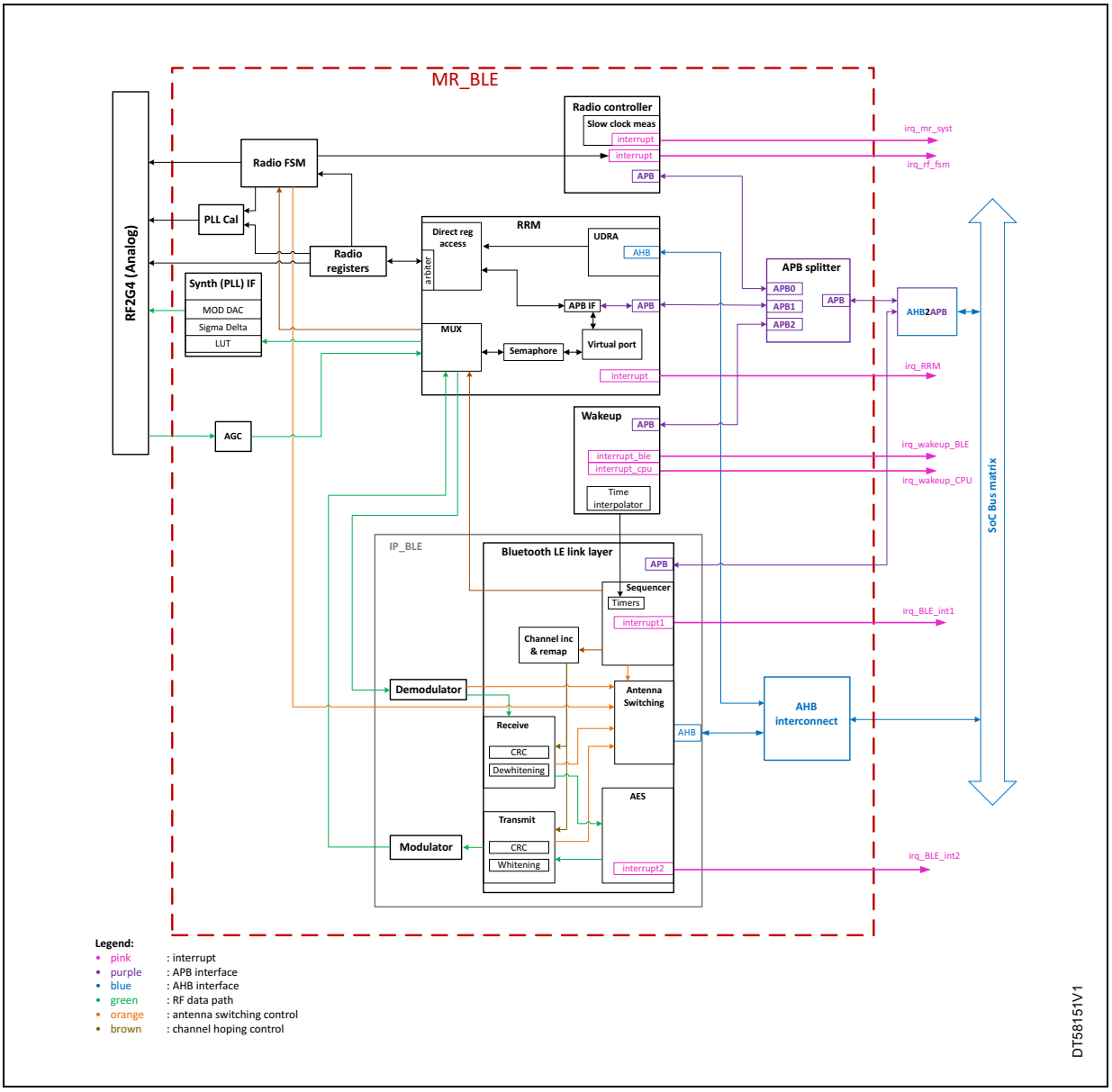
- a Bluetooth LE subsystem dedicated to Bluetooth protocol (IP_BLE block) and containing:
  - a Bluetooth LE link layer

    Note: this sub-block is an AHB master
  - a demodulator,
  - a modulator
- a RRM (radio resource manager) block containing:
  - an UDRA (unified direct register access) executing link list command in RAM to read/write radio register.

    Note: this sub-block is an AHB master
  - a direct radio register access allowing read/write access to radio registers through APB.
- A wakeup / always-on block containing:
  - a wakeup block for IP_BLE including a time interpolator and sleep request / wakeup request management and few enhanced features,
- A Radio FSM controlling the RF2G4 analog radio.
- A Radio register block (to program the analog radio parameters)
- Some small blocks used for calibrations, PLL control, RX data path interface (AGC, FIFO ADC)

The radio controller MR_BLE IP supports 2 system clock frequencies: 16 MHz and 32 MHz.

Figure 197. Radio controller MR_BLE architecture overview



25.1.2 Global scenario for Bluetooth LE protocol usage

The Bluetooth LE link layer always needs a trigger event to start a sequence. This trigger event has three possible timer sources (Wakeup timer, Timer1 or Timer2).

Basically, a Bluetooth sequence follows the steps described below:

1. The CPU needs to prepare in RAM (through different tables) all information needed for the coming transfer:
   - for radio programming (channel number, reception or transmission, calibration feature, etc.)
   - for packet to transmit or receive (contents, ADV or data, encryption, etc.)
   - for interrupt mask selection,
   - to prepare the trigger event to schedule the next sequence.
2. Then the CPU has to program the chosen timer (except if already done through the table for timer2 inside the previous sequence).
3. Once the Bluetooth LE link layer receives the trigger event from the programmed timer, it gets all information in the RAM table and launches the requested transfer.
4. At the end of the sequence (and not before), the Bluetooth LE link layer writes back in RAM tables some information, updates the status/error flags and generates an interrupt towards the CPU according to the interrupt mask programming.
5. The CPU has to treat the RAM table updated information (including packet payload after a reception) to decide what is the next wanted sequence. Then a cycle restarts from step 1.

Note: The RAM tables are located in a retention area. So the data are kept even during low power modes that switch off the digital 1.2 V power domain. This allows the Bluetooth LE link layer starting a transmission/reception while the CPU is not yet available as rebooting from a wakeup reset.

25.1.3 Miscellaneous features: RF activity monitoring

On-going TX sequence information (to control an external Power Amplifier (PA))

The radio controller MR_BLE is able to control an external Power Amplifier (located on the board to increase the Tx power) through a signal provided to the SoC to be connected directly on an external GPIO or to be managed with additional logical mechanism. The external PA can be useful to extend the TX power.

The external Power Amplifier components have quite a long latency to establish and consume a lot of current (several mA) which could lead to unlock the Radio controller MR_BLE PLL. For this reason, the control signal is anticipated as soon as the system knows it starts a TX sequence and stops as soon as possible (at the same time of the internal PA).

The Radio controller MR_BLE outputs a signal to the SoC, which is high as soon as a TX sequence is requested to the radio FSM up to the return to ACTIVE2 or less state. to be more precise, the information is provided as soon as the radio FSM is starting to treat a TX request (leaving ACTIVE2 state for EN_LDO state) and goes down as soon as the system

switches off the internal PA (e.g. leaving TX state at the end of the transmission or during EN_PA in case of PLL lock fail).

The information is available in two different ways in the STM32WB09xE:

- directly output on a GPIO (RADIO_TX_SEQUENCE)
- flags and associated interrupt available in the System Controller (SYSCFG block)

On-going RX sequence information (to control an external Low Noise Amplifier (LNA))

The Radio controller MR_BLE is able to control an external Low Noise Amplifier (located on the board) through a signal provided to the SoC to be connected directly on an external GPIO or to be managed with additional logical mechanism.

The Radio controller MR_BLE outputs a signal to the SoC, which is high as soon as a RX sequence is requested to the radio FSM up to the return to ACTIVE2 or less state.

The information is available in two different ways in the STM32WB09xE:

- directly output on a GPIO (RADIO_RX_SEQUENCE)
- flags and associated interrupt available in the System Controller (SYSCFG block)

RF activity information

An additional signal is available on device pin to inform either a RX or a TX sequence is active. This pin called RADIO_RF_ACTIVITY is a logical OR between the RADIO_TX_SEQUENCE and RADIO_RX_SEQUENCE mentioned in the two previous sub-chapters.

25.1.4 Radio controller MR_BLE IP evolution

Bluetooth LE standard 5.2 additional support

An upgrade has been done in this version of the Radio controller MR_BLE IP to allow HW adaptive behavior on encrypted packets versus the Physical Channel PDU type. Indeed, the encryption algorithm is supposed to use different masks on the header depending on the PDU type.

This upgrade impacts the StatMach word 0 with the addition of the PhysChanPDUType[1:0] bit field to help the AES block to adapt the header mask for on-the-fly encryption as follows:

00 = ACL: header mask = 0xE3 (bits 2, 3, 4 stuck at 0)

01 = CIS: header mask = 0xA3 (bits 2, 3, 4, 6 stuck at 0)

10 = BIS: header mask = 0xC3 (bits 2, 3, 4, 5 stuck at 0)

11 = RFU: header mask = 0xFF (no mask)

This means the HW encryption mode can be used on Radio controller MR_BLE for isochronous channels while only SW implementation is possible on previous versions.

LDO_TRANSFO bypass mode management versus TX power

To reach the TX power at +8dB, the LDO_TRANSFO needs to be bypassed to supply the Power Amplifier directly with the SMPS output voltage (programmed at 1.9 V). This bypass mode must be activated only during the TX (impact RX performances).

Up to now, the enable/disable action of this bypass mode has to be done by SW which is constraining in term of real time management especially during short TX/RX switch like T_IFS and T_MSS window.

A HW dynamic management of the LDO_TRANSFO bypass mode enable/disable has been added in the Radio controller MR_BLE IP:

- A new bit (TxHp) has been introduced in the StatMach to indicate if the LDO_TRANSFO bypass mode must be enabled during TX for this connection.
- If TxHp = 1, the radio FSM automatically manages the control of the bypass mode (enabled during a TX and disabled the rest of the time).

Warning: this modification implies the SW developed for previous version of the radio is no more compatible to achieve a TX power at +8dB and an upgrade is mandatory.

25.2 Interfacing with the Radio controller MR_BLE IP

The Radio controller MR_BLE IP interfaces with several blocks in the system:

- CPU through interrupts.
- RAM through AHB interface either initiated by the RRM or by the Bluetooth LE link layer.
- Power controller block and clock and reset controller block.
- Specific signals to propagate to the pads of the device to manage inside the Soc

25.2.1 Interrupt lines to the CPU

The Radio controller MR_BLEIP provides several interruption lines to the CPU, issued by different sub-blocks:

- 2 interrupt lines from the Bluetooth LE link layer,
- 2 interrupt lines from the wakeup block,
- 1 interrupt line from the RRM,
- 2 interrupt lines from the radio controller

Table 122. Radio controller MR_BLE interruptions summary

Line number on SoC NVIC	Interrupt name	Description
18	BLE_TXRX (int_BLE_irq1)	Indicate a Bluetooth sequence occurred
19	BLE_AES (int_BLE_irq2)	Indicate an AES LE privacy or manual operation ended
21	RADIO_CTRL (int_mr_syst)	Indicate the Slow clock measurement result is ready
22	MR_BLE (irq_rrm + irq_radio_fsm)	Combine information related to RRM-UDRA operations and radio FSM analog feedbacks (PLL lock and calibration)
23	CPU_WKUP (irq_wakeup_cpu)	Indicate a CPU wakeup timer match
24	BLE_WKUP (irq_wakeup_ble)	Indicate a Wakeup timer trigger occurred on Radio controller MR_BLE sequencer.

IP_BLE interrupt lines

irq_BLE_int1 (also called BLE_TXRX):

- This interrupt line is triggered when a Bluetooth sequence has been executed. It informs the CPU that status/error flags and RAM tables may have been updated,
- Mapped on CPU interrupt line 18

irq_BLE_int2 (also called BLE_AES):

- This interrupt line is dedicated to embedded AES block and combines both AES LE privacy and AES manual encryption information,
- Mapped on CPU interrupt line 19

See *Section 25.9.3: IP_BLE interrupts* for details on interrupt sources.

Wakeup block interrupt lines

irq_wakeup_ble (also called BLE_WKUP):

- This interrupt line indicates to the CPU that the IP_BLE receives a wakeup request from the Radio controller MR_BLE IP Wakeup block. The w akeup request source is the wakeup timer.
- Mapped on CPU line 24

irq_wakeup_cpu:

- This interrupt line indicates that the wakeup timer reaches the programmed value to trigger an interrupt towards the CPU (dedicated slice in the Wakeup block design in parallel of the IP_BLE slice).
- Mapped on CPU line 23

See *Section 25.13: Wakeup block* for more details.

RRM interrupt line

This interrupt line is dedicated to the RRM block. It is triggered on Semaphore and UDRA sub-block activities.

See *Section 25.6.3: Radio FSM interrupts* for more details.

Radio controller interrupt lines

irq_rf_fsm:

- This interrupt line indicates events and error detected by the radio state machine.

irq_mr_syst:

- This interrupt line is dedicated to the counter measuring the slow clock period and frequency. It indicates when the calculated values are available.

See *Section 25.7: Radio controller* for more details.

25.2.2 Interface with the RAM embedded in the SoC

The Radio controller MR_BLE IP is an AHB master on the bus matrix as the CPU or a DMA.

Two of its internal blocks access to the RAM of the system:

- the Bluetooth LE link layer to read the sequence information and Tx/Rx configuration parameters, to read the data to transmit or to write the received data in the RAM tables, to store the IQ samples during a CTE reception,
- the RRM UDRA block to get commands structure in RAM, used to program the radio registers while the CPU is not available (potentially rebooting after a low power mode).

Both internal blocks may need to access to the RAM in parallel. An arbiter is located inside the Radio controller MR_BLE.

25.2.3 Interface with the power, clock and reset controllers

The Radio controller MR_BLE IP receives several clocks from the SoC:

- the system clock that can be 16 or 32 MHz
- an always 16 MHz clock
- an always 32 MHz clock
- a slow clock (called 32 KHz clock in this document) that needs to be in the always-on power domain as it corresponds to the clock used to wake up the Radio controller MR_BLE radio link layers from sleep state.

The fast clocks (system, 16 MHz and 32 MHz) must be accurate when the radio is used (for the transmission/reception).

The Radio controller MR_BLE IP communicates with the power controller of the system to indicate when the Radio controller MR_BLE IP is OK to go to low power modes and when it requests a wakeup.

The Radio controller MR_BLE IP says it is ready to sleep when:

- no sequence is on-going (a sequence starts with a trigger event and the IP_BLE interrupt rising edge),
- no Timer1 or Timer2 selected to generate the next trigger event on the sequencer,
- no pending interrupt in the INTERRUPT1REG APB register.

The Radio controller MR_BLE IP also proposes an embedded always-on timer that may generate a programmed wakeup for the CPU.

25.3 Radio resource manager (RRM)

The radio resource manager (RRM) is a hardware block managing the radio access to one unique controller at a time.

25.3.1 Overview

The radio resource manager (RRM) is the block that manages the requests performed by the Bluetooth LE link layer of the IP_BLE and the CPU to access the radio resources. The requests pass through a semaphore and only one of the two can take control of the radio at a time. The arbitration behaves as follows:

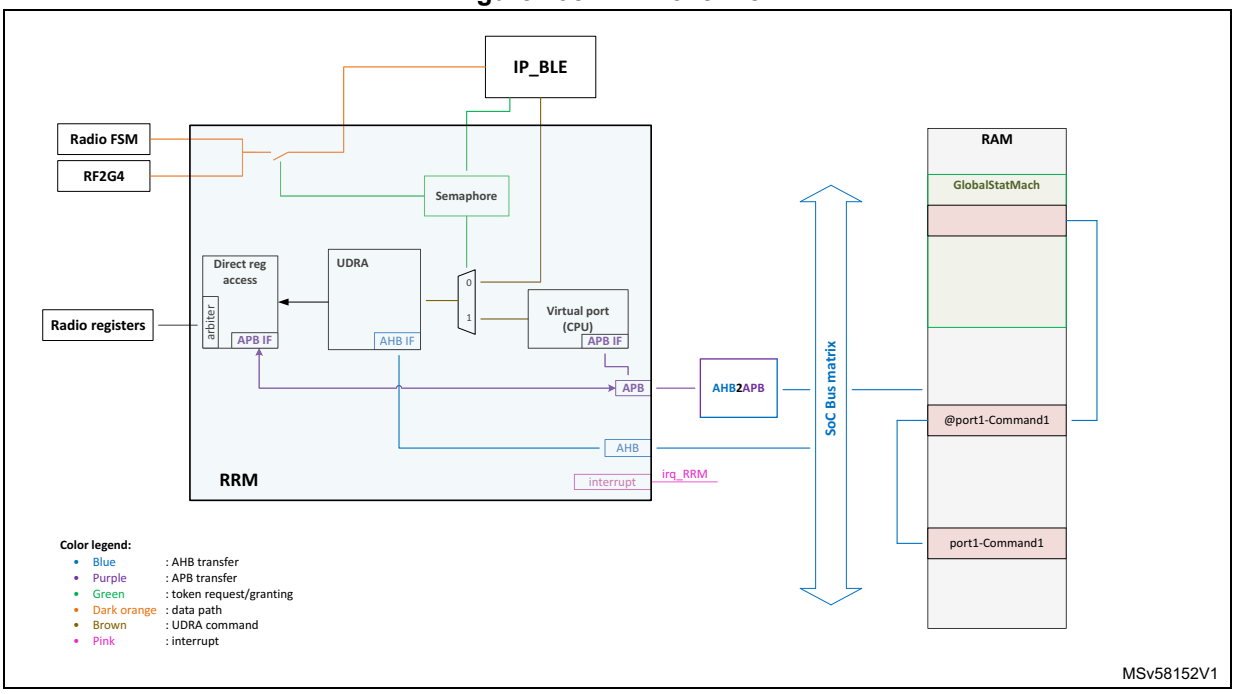
- check the priority value to choose between the IP_BLE or the CPU.
- if the same, then the arbiter eliminates the requester that has been served more recently.

The two controllers can request access to the radio resources through a dedicated port:

- Port 0 for the IP_BLE
- Port 1 for the CPU (it is a virtual port in this case)

The Bluetooth LE link layer does not have access to the radio resources until it requests a token to the RRM and the RRM grants it. For the CPU, only RRM commands require to get a token, the direct APB access to the radio registers is always possible. The token is requested by software for the CPU while it is done by hardware for the Bluetooth LE link layer each time a timer trig event starts a sequence. Nevertheless, the firmware can release the token granted by the Bluetooth LE link layer writing inside the CMDREG APB register. Once the requester has the token, its port is granted, and it can access the radio resources.

Figure 198. RRM overview



25.3.2 Semaphore

The semaphore grants the access to the radio resources control to the radio controller or to the CPU depending on the demand.

An arbitration system is in place to manage conflicts when a token request is raised simultaneously by the IP_BLE and the CPU. In this case, the arbiter:

- checks the priority programmed for the CPU Virtual Port to choose between the two requesters (IP_BLE has priority=0)
- if the same, eliminates the requester that have been served more recently.

The CPU is a virtual port and must provide the token request through an APB register bit field (see *Section 25.4: RRM registers*). The Bluetooth LE link layer uses hard wired signals to communicate with the RRM and the request is managed directly by the Bluetooth LE link layer each time a timer trig event starts a Bluetooth LE sequence.

As soon as the IP_BLE port is granted, the radio FSM block is informed by the Semaphore that the radio is about to be used. The goal is to switch off the analog as much as possible when not used and switch it on only when needed (for power consumption)

Some interruptions are linked to the Semaphore block in the RRM:

- on a port grant event,
- on a port release event,

See *Section 25.4: RRM registers* for more details.

25.3.3 UDRA

The Unified Direct Register Access block allows the software preparing some commands in a command link list located in the retention RAM. Those commands execute read from and write into the radio registers.

Some interruptions are linked to the UDRA block in the RRM:

- on a command start event,
- on a command end event.

See *Section 25.4: RRM registers* for more details.

The main goal of this block is to allow the Bluetooth LE link layer to reinitialize the radio registers after a low power mode sequence to start a RF communication while the CPU is still booting and not yet available to manage.

Note: The read command embeds some limitations. However, the radio registers can be read directly through APB by the CPU so the read command of the UDRA is useless.

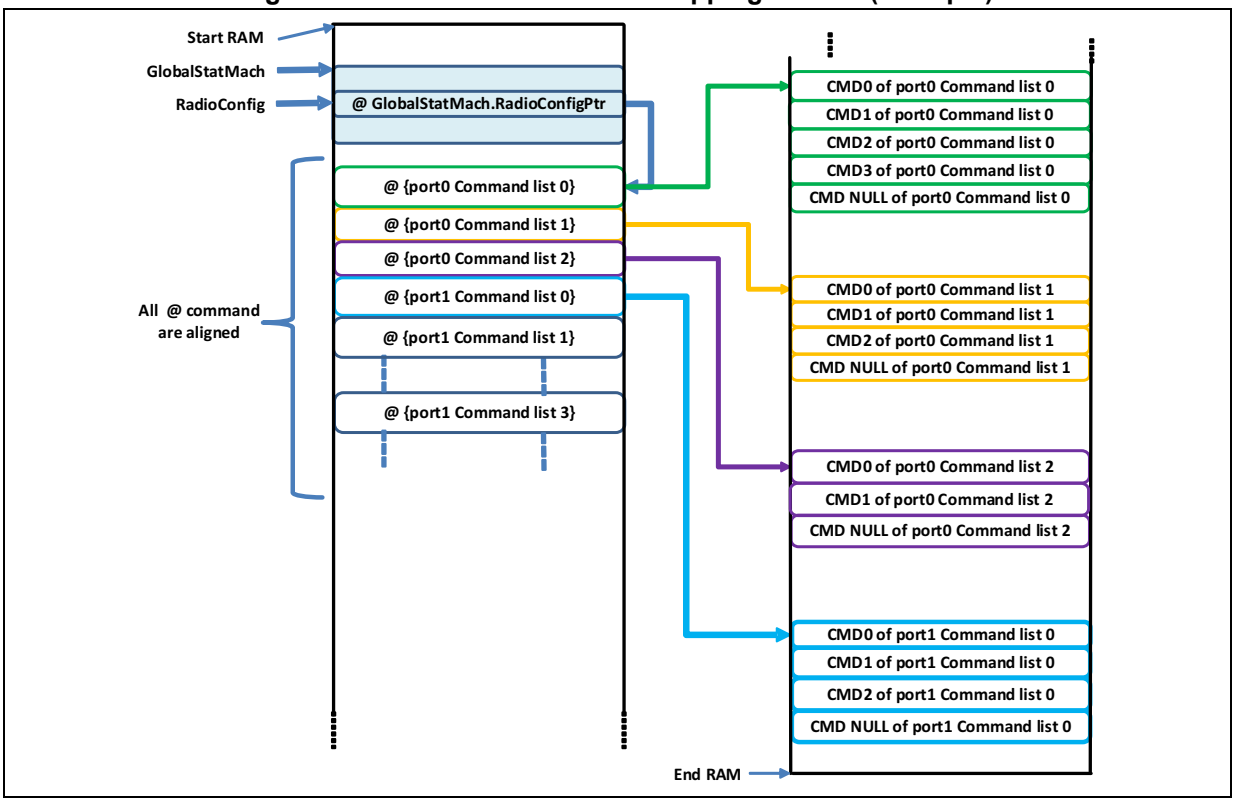
RAM link list information

The mapping in RAM for those commands is the following:

- the RadioConfigPtr field of the GlobalStatMach contains the start address of the command start list (this address must be 32-bit aligned),
- the command start list is a 32-bit elements table containing the first command addresses of each command number of each port.
- each command of each port contains some read and/or write actions on radio registers.

An overview of this indirect mapping is represented in *Figure 199*.

Figure 199. UDRA command list mapping in RAM (example)



The RadioConfigPtr value is loaded by the RRM-UDRA automatically when the Radio controller MR_BLE IP reset is released.

If the software did not initialize this RAM address supposed to point on the command_start_list address before this first automatic load, a “reload pointer” command is available by writing 1 in UDRA_CTRL0[0] APB register (this bit is auto-cleared immediately).

Note: The RadioConfigPtr pointer value loaded and used by the RRM-UDRA block can be read in the UDRA_RADIO_CFG_PTR APB register.

For the Radio controller MR_BLE IP, the port mapping has been defined as follow:

- 2 ports (port0=IP_BLE, port1=VP_CPU)
- port0 supports 3 commands (command 0/1/2)
- port1 supports 4 commands (command 0/1/2/3)

This leads to the command list given in *Figure 123*:

Table 123. Command start list details

Address in RAM	Meaning	Comments
@RadioConfigPtr(value) + 0x00	port0→command0 base address	Command requested by the Bluetooth LE link layer on wakeup timer trigger event if RadioComListEna bit = 1 in on-going Stat-Mach.
@RadioConfigPtr(value) + 0x04	port0→command1 base address	Command requested by the Bluetooth LE link layer on Timer1 trigger event if Radio-ComListEna bit = 1 in on-going StatMach
@RadioConfigPtr(value) + 0x08	port0→command2 base address	Command requested by the Bluetooth LE link layer on Timer2 trigger event if Radio-ComListEna bit = 1 in on-going StatMach
@RadioConfigPtr(value) + 0x0C	port1→command0 base address	VP_CPU: if the software needs to use a RRM-UDRA command to access to the radio register instead of a direct access through APB.
@RadioConfigPtr(value) + 0x10	port1→command1 base address	VP_CPU: if the software needs to use a second RRM-UDRA command to access to the radio register instead of a direct access through APB
@RadioConfigPtr(value) + 0x14	port1→command2 base address	VP_CPU: if the software needs to use a third RRM-UDRA command to access to the radio register instead of a direct access through APB
@RadioConfigPtr(value) + 0x18	port1→command3 base address	VP_CPU: if the software needs to use a fourth RRM-UDRA command to access to the radio register instead of a direct access through APB

UDRA command format in RAM

The write and read commands format is described in *Table 124*.

Note that only one radio register address is entered for a write or a read. Then, if the number of data to write/read is more than one, the address is incremented automatically by 1.

Table 124. UDRA command format in RAM

Byte number	Address in RAM	Byte value	Description
1	command_base_addr	0x--	bit7: 0=write / 1=read bit[6:0] = number of data to write or to read. n = number of data for the example in this table.
2	command_base_addr+1	8-bit address	Address of a Radio register following the 8-bit address mapping (see *Section 25.5: Radio registers*)
3	command_base_addr+2	1st data	If write command: write first 8-bit data to be written. If read command: location where the first 8-bit read data is available.
4	command_base_addr+3	2nd data	Optional (depends on number of data to write/read). If write command: write second 8-bit data to be written. If read command: location where the second 8-bit read data is available.
…			
n+2	command_base_addr+(n+1)	nth data	Optional (depends on number of data to write/read). If write command: write nth 8-bit data to be written. If read command: location where the nth 8-bit read data is available.
n+3	command_base_addr+n+2	0x--	Optional: possible to chain other commands. bit7: 0=write / 1=read bit[6:0] = number of data to write or to read.
n+4	command_base_addr+n+3	8-bit address	Address of a Radio register following the 8-bit address mapping (see *Section 25.5: Radio registers*)
n+5	command_base_addr+n+4	1st data	If write command: write first 8-bit data to be written. If read command: location where the first 8-bit read data is available.
…			
last	command_base_addr+last-1	0x00 / 0x80	MANDATORY. The null command (command with null length) must be added at the end of the command list. This is needed by the state machines of the UDRA to be informed they reach the end of the list.

Note: If any error information is put in the RAM command list (bad command number, lack of null command, and so on), the RRM-UDRA does not returns to an IDLE state and cannot accept new command until a reset is done on the full Radio controller MR_BLE IP.

Basic example: Write AAC0_DIG_ENG=0x12 and AAC1_DIG_ENG=0x34 (grouped registers) through port1.command0:

@port1.command0_addr = 0x02; Write 2 data

@port1.command0_addr+1 = 0x AAC0_DIG_ENG_ADDR;

@port1.command0_addr+2 = 0x12; 1st data to write in AAC0_DIG_ENG

@port1.command0_addr+3 = 0x34; 2nd data to write in AAC1_DIG_ENG

@port1.command0_addr+4 = 0x00; null command

At the end of command execution, the 2 radio registers have been modified with new value.

25.3.4 Direct register access

The Direct Register access block allows the software accessing the radio registers directly through an APB access.

To do a direct read or write access to the radio registers, the software just has to read/write them through APB at address mapping described in section *Section 25.5: Radio registers*.

Note: The radio registers are 8-bit only so the APB registers bit field[31:8] part is padded with 0.

A 8-bit address mapping column is provided to be used in RRM UDRA command list as address of the radio register in this specific case.

An internal arbiter manages the case of concurrent accesses on radio registers by both UDRA (executing a command) and Direct Register Access block (on a CPU read/write APB request). The arbitration is round-robin.

Important recommendation:

The software must not write some radio registers through direct APB access if they are also modified through commands in RAM (through UDRA block). In this case, there is a risk of multi drivers in parallel and to lose coherency (no way to know which requester wrote the last).

25.4 RRM registers

The RRM registers are accessible through the APB interface.

All non-listed addresses must be considered as RESERVED.

The **RRM_BLOCKBaseAddress** keyword used for all the register base address information corresponds to the RRM registers base address in the SoC when integrating the IP.

RRM_BLOCKBaseAddress is 0x6000_1400 in the STM32WB09xE product.

Table 125. RRM register list

Offset	Register Name	Register Title	Reset
0x0010	UDRA_CTRL0	UDRA_CTRL0 register	0x0000 0000
0x0014	UDRA_IRQ_ENABLE	UDRA_IRQ_ENABLE register	0x0000 0000
0x0018	UDRA_IRQ_STATUS	UDRA_IRQ_STATUS register	0x0000 0000
0x001C	UDRA_RADIO_CFG_PTR	UDRA_RADIO_CFG_PTR register	0x0000 0000
0x0020	SEMA_IRQ_ENABLE	SEMA_IRQ_ENABLE register	0x0000 0000
0x0024	SEMA_IRQ_STATUS	SEMA_IRQ_STATUS register	0x0000 0000
0x0028	BLE_IRQ_ENABLE	BLE_IRQ_ENABLE register	0x0000 0000
0x002C	BLE_IRQ_STATUS	BLE_IRQ_STATUS register	0x0000 0000
0x0060	VP_CPU_CMD_BUS	VP_CPU_CMD_BUS register	0x0000 0000
0x0064	VP_CPU_SEMA_BUS	VP_CPU_SEMA_BUS register	0x0000 0000
0x0068	VP_CPU_IRQ_ENABLE	VP_CPU_IRQ_ENABLE register	0x0000 0000
0x006C	VP_CPU_IRQ_STATUS	VP_CPU_IRQ_STATUS register	0x0000 0000

25.4.1 UDRA_CTRL0 register (UDRA_CTRL0)

Address offset: 0x0010

Reset value: 0x0000 0000 (0xFFFF FFFF)

UDRA control register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RELOAD_RDCFGPTR
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **RELOAD_RDCFGPTR**: reload the radio configuration pointer from RAM.
This bit is auto cleared by hardware.

25.4.2 UDRA_IRQ_ENABLE register (UDRA_IRQ_ENABLE)

Address offset: 0x0014

Reset value: 0x0000 0000 (0xFFFF FFFF)

UDRA Interrupt mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD_END	CMD_START	RADIO_CFG_PTR_RELOADED
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CMD_END**: UDRA interrupt enable (command end)

Bit 1 **CMD_START**: UDRA interrupt enable (command start)

Bit 0 **RADIO_CFG_PTR_RELOADED**: UDRA interrupt enable (reload radio config pointer)

25.4.3 UDRA_IRQ_STATUS register (UDRA_IRQ_STATUS)

Address offset: 0x0018

Reset value: 0x0000 0000 (0xFFFF FFFF)

DEUDRA Interrupt status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD_END	CMD_START	RADIO_CFG_PTR_RELOADED
													rc_w1	rc_w1	rc_w1

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CMD_END**: On read, returns the UDRA command end interrupt status
Write '1' to clear IRQ status bit.
Note: this flag is located at global UDRA level and is raised only at the beginning of the command list execution (not raised again at start of all sub-commands chained in the command number under execution before the NULL command)

Bit 1 **CMD_START**: On read, returns the UDRA command start interrupt status.
Write '1' to clear IRQ status bit.

Bit 0 **RADIO_CFG_PTR_RELOADED**: On read, returns the UDRA reload radio configuration pointer interrupt status.
Write '1' to clear IRQ status bit.

25.4.4 UDRA_RADIO_CFG_PTR register (UDRA_RADIO_CFG_PTR)

Address offset: 0x001C

Reset value: 0x0000 0000 (0xFFFF FFFF)

UDRA radio configuration pointer register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RADIO_CONFIG_ADDRESS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RADIO_CONFIG_ADDRESS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RADIO_CONFIG_ADDRESS[31:0]**: UDRA radio configuration address.
This field contains the value contained by RadioConfigPtr bit field in GlobalStatMach RAM table when the MR_BLE51 IP exits the reset state.
This field is updated after a reload configuration pointer command.

25.4.5 SEMA_IRQ_ENABLE register (SEMA_IRQ_ENABLE)

Address offset: 0x0020

Reset value: 0x0000 0000 (0xFFFF FFFF)

Semaphore Interrupt mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK	LOCK
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UNLOCK**: semaphore unlocked (=no port selected) interrupt enable

Bit 0 **LOCK**: semaphore locked (= one port granted) interrupt enable

25.4.6 SEMA_IRQ_STATUS register (SEMA_IRQ_STATUS)

Address offset: 0x0024

Reset value: 0x0000 0000 (0xFFFF FFFF)

Semaphore Interrupt status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK	LOCK
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UNLOCK**: On read, returns the semaphore unlocked interrupt status.
Note: this flag reacts only on Bluetooth LE link layer token release but not on VP_CPU token release (which is useless as the SW is responsible to the action by clearing the take_req bit.
Write '1' to clear this IRQ status bit.

Bit 0 **LOCK**: On read, returns the semaphore locked interrupt status.
Write '1' to clear this IRQ status bit.

25.4.7 BLE_IRQ_ENABLE register (BLE_IRQ_ENABLE)

Address offset: 0x0028

Reset value: 0x0000 0000 (0xFFFF FFFF)

IP_BLE IT mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_CMD_END	PORT_CMD_START	Res.	PORT_RELEASE	PORT_GRANT
											rw	rw		rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **PORT_CMD_END**: IP_BLE Port command end interrup enable

Bit 3 **PORT_CMD_START**: IP_BLE Port command start interrup enable

Bit 2 Reserved, must be kept at reset value.

Bit 1 **PORT_RELEASE**: IP_BLE Port release interrupt enable

Bit 0 **PORT_GRANT**: IP_BLE Port grant interrupt enable

25.4.8 BLE_IRQ_STATUS register (BLE_IRQ_STATUS)

Address offset: 0x002C

Reset value: 0x0000 0000 (0xFFFF FFFF)

IP_BLE IT status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD_END	CMD_START	Res.	PORT_RELEASE	PORT_GRANT
											rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMD_END**: IP_BLE hardware port command end interrupt status.
When read:
0: The IP_BLE port command requested by the Bluetooth LE link layer is not completed.
1: The IP_BLE port command requested by the Bluetooth LE link layer is completed.
Note: this flag is raised only when the UDRA reaches the NULL command (not as CMD_START)
Write '1' to clear this IRQ status bit.

Bit 3 **CMD_START**: IP_BLE hardware port command start interrupt status.
When read:
0: The IP_BLE port command requested by the Bluetooth LE link layer is not started.
1: The IP_BLE port command requested by the Bluetooth LE link layer is started.
Note: this flag is raised at the beginning of any chained sub-command inside the command number under execution (so can be raised several times if more than one command before the NULL command)
Write '1' to clear this IRQ status bit.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **PORT_RELEASE**: IP_BLE hardware port released interrupt status.
When read:
0: The IP_BLE controller has not been released (due to take_req=1'b1)
1: The IP_BLE controller has been released by the semaphore due to take_req=1'b0 (requested by Bluetooth LE link layer).
Write '1' to clear this IRQ status bit.

Bit 0 **PORT_GRANT**: IP_BLE hardware port granted interrupt status.
0: the IP_BLE port request to semaphore is not granted.
1: the Bluetooth LE link layer request to take the semaphore is granted: the Radio registers access and the Radio TX and the Radio RX data path are selected for that controller. The port stays granted as long as it requests the token.
Write '1' to clear this IRQ status bit.

Note: The Bluetooth LE link layer receives that information directly by hardware wires and manages the sequence through them. The interrupt mechanism is there in case the CPU needs to monitor the activity between the Bluetooth LE link layer and the RRM block.

25.4.9 VP_CPU_CMD_BUS register (VP_CPU_CMD_BUS)

Address offset: 0x0060

Reset value: 0x0000 0000 (0xFFFF FFFF)

CPU Virtual Port Command bus register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMMAND_REQ	COMMAND[2:0]		
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **COMMAND_REQ**: CPU Virtual port command request.
0: the RRM command request is released
1: request a command to the RRM-UDRA block.
This bit is cleared by HW once the command is ended.

Bits 2:0 **COMMAND[2:0]**: command number

25.4.10 VP_CPU_SEMA_BUS register (VP_CPU_SEMA_BUS)

Address offset: 0x0064

Reset value: 0x0000 0000 (0xFFFF FFFF)

CPU Virtual Port semaphore bus register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAKE_REQ	TAKE_PRIO[2:0]		
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **TAKE_REQ**: Semaphore token request.

0: The CPU virtual port releases the semaphore or does not request to take the RRM semaphore.

1: The CPU virtual port requests to take or to keep the RRM semaphore.

Bits 2:0 **TAKE_PRIO[2:0]**: semaphore priority: priority value (between 0 and 7) of the take request. The higher the value, the higher priority is the request.

25.4.11 VP_CPU_IRQ_ENABLE register (VP_CPU_IRQ_ENABLE)

Address offset: 0x0068

Reset value: 0x0000 0000 (0xFFFF FFFF)

CPU Virtual Port Interrupt mask register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_CMD_END	PORT_CMD_START	Res.	PORT_RELEASE	PORT_GRANT
											rw	rw		rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **PORT_CMD_END**: CPU virtual port command end interrup enable

Bit 3 **PORT_CMD_START**: CPU virtual port command start interrup enable

Bit 2 Reserved, must be kept at reset value.

Bit 1 **PORT_RELEASE**: CPU virtual port release interrupt enable

Bit 0 **PORT_GRANT**: CPU virtual port grant interrupt enable

25.4.12 VP_CPU_IRQ_STATUS register (VP_CPU_IRQ_STATUS)

Address offset: 0x006C

Reset value: 0x0000 0000 (0xFFFF FFFF)

CPU Virtual Port Interrupt status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD_END	CMD_START	Res.	PORT_RELEASE	PORT_GRANT
											rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMD_END**: CPU virtual port command end interrupt status.
When read:
0: no command requested by the CPU virtual port (port1) is completed.
1: a command requested by the CPU virtual port (port1) is completed.
Note: this flag is raised only when the UDRA reaches the NULL command (not as CMD_START)
Write '1' to clear this IRQ status bit.

Bit 3 **CMD_START**: CPU virtual port command start interrupt status.
When read:
0: The command requested by the CPU virtual port (port1) is not started.
1: The command requested by the CPU virtual port (port1) is started
Note: this flag is raised at the beginning of any chained command inside the command number under execution (so can be raised several times if more than one command before the NULL command)
Write '1' to clear this IRQ status

Bit 2 Reserved, must be kept at reset value.

Bit 1 **PORT_RELEASE**: virtual port released interrupt status.
0: The CPU virtual port has not been released (due to TAKE_REQ=1'b1)
1: The CPU virtual port has been released by the semaphore due to TAKE_REQ=1'b0 (requested by CPU virtual port).
Write '1' to clear this IRQ status bit.

Bit 0 **PORT_GRANT**: CPU virtual port granted interrupt status.
0: The CPU virtual port token request is not granted.
1: The CPU virtual port token request is granted by the semaphore:
- the Radio registers access through UDRA command is possible for that port (direct APB access is not concerned, always accessible),
- prevents the Bluetooth LE link layer to have access to the Radio TX and the Radio RX data path.
Write '1' to clear this IRQ status bit.

25.5 Radio registers

The Radio registers are 8-bit registers mainly used to control the RF2G4 analog IP and the Radio FSM. They also allow taking control for validation/test purposes.

They can be accessed through two different mappings:

- as 32-bit APB registers (address incremented by 4 between each register) through RRM Direct Access interface
  - this mapping is used by the CPU
- as 8-bit register (address incremented by 1 between each register) through RRM UDRA command list in RAM
  - this mapping is used by the RRM

Caution: The Radio registers are used to control the analog and few modulator/demodulator features. They are not supposed to be modified directly by the user. The potential modifications are provided by STMicroelectronics in the SDK boot code.

The **RF_REG_BLOCKBaseAddress** keyword used for all the register base address information corresponds to the Radio registers base address decided by the SoC when integrating the IP.

RF_REG_BLOCKBaseAddress is 0x6000_1500 in the STM32WB09xE product.

Table 126. RADIO register list

Address offset		Register Name	Register Title	Reset
8-bit	APB			
0x00	0x0000	AA0_DIG_USR	AA0_DIG_USR register	0x0000 00D6
0x01	0x0004	AA1_DIG_USR	AA1_DIG_USR register	0x0000 00BE
0x02	0x0008	AA2_DIG_USR	AA2_DIG_USR register	0x0000 0089
0x03	0x000C	AA3_DIG_USR	AA3_DIG_USR register	0x0000 008E
0x04	0x0010	DEM_MOD_DIG_USR	DEM_MOD_DIG_USR register	0x0000 0026
0x05	0x0014	RADIO_FSM_USR	RADIO_FSM_USR register	0x0000 0004
0x06	0x0018	PHYCTRL_DIG_USR	PHYCTRL_DIG_USR register	0x0000 0000
0x12	0x0048	AFC1_DIG_ENG	AFC1_DIG_ENG register	0x0000 0044
0x15	0x0054	CR0_DIG_ENG	CR0_DIG_ENG register	0x0000 0044
0x1A	0x0068	CR0_LR	CR0_LR register	0x0000 0066
0x1B	0x006C	VIT_CONF_DIG_ENG	VIT_CONF_DIG_ENG register	0x0000 0000
0x21	0x0084	LR_PD_THR_DIG_ENG	LR_PD_THR_DIG_ENG register	0x0000 0050
0x22	0x0088	LR_RSSI_THR_DIG_ENG	LR_RSSI_THR_DIG_ENG register	0x0000 001B
0x23	0x008C	LR_AAC_THR_DIG_ENG	LR_AAC_THR_DIG_ENG register	0x0000 0038
0x2A	0x00A8	SYNTHCAL0_DIG_ENG	SYNTHCAL0_DIG_ENG register	0x0000 0000
0x3C	0x00F0	DTB5_DIG_ENG	DTB5_DIG_ENG register	0x0000 0000
0x52	0x0148	RXADC_ANA_USR	RXADC_ANA_USR register	0x0000 001B
0x55	0x0154	LDO_ANA_ENG	LDO_ANA_ENG register	0x0000 0000

Table 126. RADIO register list (continued)

Address offset		Register Name	Register Title	Reset
8-bit	APB			
0x5D	0x0174	CBIAS0_ANA_ENG	CBIAS0_ANA_ENG register	0x0000 0088
0x5E	0x0178	CBIAS1_ANA_ENG	CBIAS1_ANA_ENG register	0x0000 0000
0x60	0x0180	SYNTHCAL0_DIG_OUT	SYNTHCAL0_DIG_OUT register	0x0000 0000
0x61	0x0184	SYNTHCAL1_DIG_OUT	SYNTHCAL1_DIG_OUT register	0x0000 0001
0x62	0x0188	SYNTHCAL2_DIG_OUT	SYNTHCAL2_DIG_OUT register	0x0000 0040
0x63	0x018C	SYNTHCAL3_DIG_OUT	SYNTHCAL3_DIG_OUT register	0x0000 0000
0x64	0x0190	SYNTHCAL4_DIG_OUT	SYNTHCAL4_DIG_OUT register	0x0000 0018
0x65	0x0194	SYNTHCAL5_DIG_OUT	SYNTHCAL5_DIG_OUT register	0x0000 0007
0x66	0x0198	FSM_STATUS_DIG_OUT	FSM_STATUS_DIG_OUT register	0x0000 0000
0x69	0x01A4	RSSI0_DIG_OUT	RSSI0_DIG_OUT register	0x0000 0008
0x6A	0x01A8	RSSI1_DIG_OUT	RSSI1_DIG_OUT register	0x0000 0008
0x6B	0x01AC	AGC_DIG_OUT	AGC_DIG_OUT register	0x0000 0000
0x6C	0x01B0	DEMOD_DIG_OUT	DEMOD_DIG_OUT register	0x0000 0000
0x6F	0x01BC	AGC2_ANA_TST	AGC2_ANA_TST register	0x0000 0000
0x70	0x01C0	AGC0_DIG_ENG	AGC0_DIG_ENG register	0x0000 004A
0x71	0x01C4	AGC1_DIG_ENG	AGC1_DIG_ENG register	0x0000 0084
0x7A	0x01E8	AGC10_DIG_ENG	AGC10_DIG_ENG register	0x0000 0000
0x7B	0x01EC	AGC11_DIG_ENG	AGC11_DIG_ENG register	0x0000 0010
0x7C	0x01F0	AGC12_DIG_ENG	AGC12_DIG_ENG register	0x0000 0020
0x7D	0x01F4	AGC13_DIG_ENG	AGC13_DIG_ENG register	0x0000 0030
0x7E	0x01F8	AGC14_DIG_ENG	AGC14_DIG_ENG register	0x0000 0038
0x7F	0x01FC	AGC15_DIG_ENG	AGC15_DIG_ENG register	0x0000 0039
0x80	0x0200	AGC16_DIG_ENG	AGC16_DIG_ENG register	0x0000 003A
0x81	0x0204	AGC17_DIG_ENG	AGC17_DIG_ENG register	0x0000 003B
0x82	0x0208	AGC18_DIG_ENG	AGC18_DIG_ENG register	0x0000 003C
0x83	0x020C	AGC19_DIG_ENG	AGC19_DIG_ENG register	0x0000 003D
0x89	0x0224	RXADC_HW_TRIM_OUT	RXADC_HW_TRIM_OUT register	0x0000 001B
0x8A	0x0228	CBIAS0_HW_TRIM_OUT	CBIAS0_HW_TRIM_OUT register	0x0000 0088
0x8C	0x0230	AGC_HW_TRIM_OUT	AGC_HW_TRIM_OUT register	0x0000 0006
0x90	0x0240	ANTSW0_DIG_USR	ANTSW0_DIG_USR register	0x0000 001C
0x91	0x0244	ANTSW1_DIG_USR	ANTSW1_DIG_USR register	0x0000 000B
0x92	0x0248	ANTSW2_DIG_USR	ANTSW2_DIG_USR register	0x0000 0029
0x93	0x024C	ANTSW3_DIG_USR	ANTSW3_DIG_USR register	0x0000 0023

25.5.1 AA0_DIG_USR register (AA0_DIG_USR)

Address offset: 0x0000

Reset value: 0x0000 00D6 (0xFFFF FFFF)

access address register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA_7_0[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **AA_7_0[7:0]**: Least significant byte of the Bluetooth LE Access Address code

This register is (over)written by the sequencer during 2nd INIT step with the StatMach.accaddr[7:0] bit field.

25.5.2 AA1_DIG_USR register (AA1_DIG_USR)

Address offset: 0x0004

Reset value: 0x0000 00BE (0xFFFF FFFF)

access address register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA_15_8[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **AA_15_8[7:0]**: Next byte of the Bluetooth LE Access Address code.

This register is (over)written by the sequencer during the 2nd INIT step with the StatMach.accaddr[15:8] bit field.

25.5.3 AA2_DIG_USR register (AA2_DIG_USR)

Address offset: 0x0008

Reset value: 0x0000 0089 (0xFFFF FFFF)

access address register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA_23_16[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **AA_23_16[7:0]**: Next byte of the Bluetooth LE Access Address code

This register is (over)written by the sequencer during 2nd INIT step with the Stat-Mach.accaddr[23:16] bit field.

25.5.4 AA3_DIG_USR register (AA3_DIG_USR)

Address offset: 0x000C

Reset value: 0x0000 008E (0xFFFF FFFF)

access address register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA_31_24[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **AA_31_24[7:0]**: Most significant byte of the Bluetooth LE Access Address code.

This register is (over)written by the sequencer during 2nd INIT step with the StatMach.accaddr[31:24] bit field.

25.5.5 DEM_MOD_DIG_USR register (DEM_MOD_DIG_USR)

Address offset: 0x0010

Reset value: 0x0000 0026 (0xFFFF FFFF)

MOD DEM DIG USR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHANNEL_NUM[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:1 **CHANNEL_NUM[6:0]**: Index for internal lock up table in which the synthesizer setup is contained.

Formula for programmed frequency is: 2402 + (CHANNEL_NUM x 2)

Default value (0x13=19) corresponds to the Bluetooth LE RF channel 19: 2402 + (19 x 2) = 2440 MHz.

This bit field is (over) written by the sequencer during the1st INIT. The value copied here is the output of the channel Incr and Hoping hardware block.

Note: This bit field is used to generate the physical frequency on the antenna.

Bit 0 Reserved, must be kept at reset value.

25.5.6 RADIO_FSM_USR register (RADIO_FSM_USR)

Address offset: 0x0014

Reset value: 0x0000 0004 (0xFFFF FFFF)

RADIO_FSM_USR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PA_POWER[4:0]					EN_CALIB_SYNTH	EN_CALIB_CBP	Res.
								rw	rw	rw	rw	rw	rw	rw	

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PA_POWER[4:0]**: PA Power coefficient.
This bit is (over) written by the sequencer with the StatMach.PAPower bit field during the 1st INIT step.

Bit 2 **EN_CALIB_SYNTH**: SYNTH calibration enable bit.
This bit is (over) written by the sequencer with the TxRxPack.CalReq bit during the 1st INIT step.

Bit 1 **EN_CALIB_CBP**: CBP calibration enable bit.
This bit is (over) written by the sequencer with the TxRxPack.CalReq bit during the 1st INIT step.
Note: Both EN_CALIB_xx must be set or reset together (mixed configuration not recommended)

Bit 0 Reserved, must be kept at reset value.

25.5.7 PHYCTRL_DIG_USR register (PHYCTRL_DIG_USR)

Address offset: 0x0018

Reset value: 0x0000 0000 (0xFFFF FFFF)

PHYCTRL_DIG_USR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXTXPHY[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **RXTXPHY[2:0]**: RXTXPHY selection.

This bit field is (over) written by the sequencer during the 1st INIT using the StatMach.RxPhy[2:0] or StatMach.TxPhy[2:0], depending if the transfer is a reception or a transmission.

- 000: uncoded PHY 1Mb/s
- 001: uncoded PHY 2Mb/s
- 100: coded PHY S=8 1Mb/s
- 110: coded PHY S=2 1Mb/s

25.5.8 AFC1_DIG_ENG register (AFC1_DIG_ENG)

Address offset: 0x0048

Reset value: 0x0000 0044 (0xFFFF FFFF)

AFC1_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFC_DELAY_BEFORE[3:0]				AFC_DELAY_AFTER[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **AFC_DELAY_BEFORE[3:0]**: Set the decay factor of the AFC loop before Access Address detection

Bits 3:0 **AFC_DELAY_AFTER[3:0]**: Set the decay factor of the AFC loop after Access Address detection

25.5.9 CR0_DIG_ENG register (CR0_DIG_ENG)

Address offset: 0x0054

Reset value: 0x0000 0044 (0xFFFF FFFF)

CR0_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR_GAIN_BEFORE[3:0]				CR_GAIN_AFTER[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CR_GAIN_BEFORE[3:0]**: Set the gain of the clock recovery loop before Access Address detection to the value

2^(-CR_GAIN_BEFORE)

Bits 3:0 **CR_GAIN_AFTER[3:0]**: Set the gain of the clock recovery loop before Access Address detection to the value

2^(-CR_GAIN_AFTER)

25.5.10 CR0_LR register (CR0_LR)

Address offset: 0x0068

Reset value: 0x0000 0066 (0xFFFF FFFF)

CR0_LR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR_LR_GAIN_BEFORE[3:0]				CR_LR_GAIN_AFTER[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CR_LR_GAIN_BEFORE[3:0]**: Set the gain of the clock recovery loop before Access Address detection to the value 2^(-CR_LR_GAIN_ BEFORE) when the coded PHY is in use

Bits 3:0 **CR_LR_GAIN_AFTER[3:0]**: Set the gain of the clock recovery loop after Access Address detection to the value 2^(-CR_LR_GAIN_ AFTER) when the coded PHY is in use

25.5.11 VIT_CONF_DIG_ENG register (VIT_CONF_DIG_ENG)

Address offset: 0x006C

Reset value: 0x0000 0000 (0xFFFF FFFF)

VIT_CONF_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPARE[5:0]						Res.	VIT_EN
								rw	rw	rw	rw	rw	rw		rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:2 **SPARE[5:0]**: spare

Bit 1 Reserved, must be kept at reset value.

Bit 0 **VIT_EN**: Viterbi enable
0: Viterbi is disabled
1: Viterbi is enabled

25.5.12 LR_PD_THR_DIG_ENG register (LR_PD_THR_DIG_ENG)

Address offset: 0x0084

Reset value: 0x0000 0050 (0xFFFF FFFF)

LR_PD_THR_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LR_PD_THR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LR_PD_THR[7:0]**: preamble detect threshold value

25.5.13 LR_RSSI_THR_DIG_ENG register (LR_RSSI_THR_DIG_ENG)

Address offset: 0x0088

Reset value: 0x0000 001B (0xFFFF FFFF)

LR_RSSI_THR_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LR_RSSI_THR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LR_RSSI_THR[7:0]**: RSSI or peak threshold value

25.5.14 LR_AAC_THR_DIG_ENG register (LR_AAC_THR_DIG_ENG)

Address offset: 0x008C

Reset value: 0x0000 0038 (0xFFFF FFFF)

LR_AAC_THR_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LR_AAC_THR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LR_AAC_THR[7:0]**: address coded correlation threshold

25.5.15 SYNTHCAL0_DIG_ENG register (SYNTHCAL0_DIG_ENG)

Address offset: 0x00A8

Reset value: 0x0000 0000 (0xFFFF FFFF)

SYNTHCAL0_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNTH_IF_FREQ_CAL[1:0]		Res.	Res.	SYNTHCAL_DEBUG_BUS_SEL[3:0]			
								rw	rw			rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SYNTH_IF_FREQ_CAL[1:0]**: Define the frequency applied on the PLL during calibration phase

- 00 (default): calibration is done between RX and TX frequencies (at freq_channel –0.8 MHz)
- 01: calibration is done at TX frequency (at freq_channel)
- 10: calibration is done at RX frequency (at freq_channel –1.6 MHz)
- 11Reserved

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **SYNTHCAL_DEBUG_BUS_SEL[3:0]**: for Debug purpose

Program 0xC to get the PLL calibration reason in SYNTHCAL3_DIG_OUT

25.5.16 DTB5_DIG_ENG register (DTB5_DIG_ENG)

Address offset: 0x00F0

Reset value: 0x0000 0000 (0xFFFF FFFF)

DTB5_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_SELECTED_0	PORT_SELECTED_EN	INITIALIZE	RX_ACTIVE	TX_ACTIVE	RXTX_START_SEL
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **PORT_SELECTED_0**: force port_selected[0] signal

Bit 4 **PORT_SELECTED_EN**: enable port selection

Bit 3 **INITIALIZE**: Force INITIALIZE signal (emulate a token request of the IP_BLE)

Bit 2 **RX_ACTIVE**: Force RX_ACTIVE signal

Bit 1 **TX_ACTIVE**: Force TX_ACTIVE signal

Bit 0 **RXTX_START_SEL**: Enable the possibility to control some signals by the other register bits instead of system design.

0: the Radio FSM is controlled dynamically by the signals generated by the RRM and sequencer

1: the Radio FSM is controlled by the bits of this register.

25.5.17 RXADC_ANA_USR register (RXADC_ANA_USR)

Address offset: 0x0148

Reset value: 0x0000 001B (0xFFFF FFFF)

RXADC_ANA_USR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXADC_DELAYTRIM_Q_TST_SEL	RXADC_DELAYTRIM_I_TST_SEL	RFD_RXADC_DELAYTRIM_Q[2:0]			RFD_RXADC_DELAYTRIM_I[2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RXADC_DELAYTRIM_Q_TST_SEL**: Enable the SW overload on RXADX delay trimming
0: trimming applied on the analog block are the HW loaded ones,
1: trimming applied on the analog block are provided by the RFD_RXADC_DELAYTRIM_Q[2:0] bit field (SW values).

Bit 6 **RXADC_DELAYTRIM_I_TST_SEL**: Enable the SW overload on RXADX delay trimming
0: trimming applied on the analog block are the HW loaded ones,
1: trimming applied on the analog block are provided by the RFD_RXADC_DELAYTRIM_I[2:0] bit field (SW values).

Bits 5:3 **RFD_RXADC_DELAYTRIM_Q[2:0]**: ADC loop delay control bits for Q channel to apply when SW overload is enabled

Bits 2:0 **RFD_RXADC_DELAYTRIM_I[2:0]**: ADC loop delay control bits for I channel to apply when SW overload is enabled

25.5.18 LDO_ANA_ENG register (LDO_ANA_ENG)

Address offset: 0x0154

Reset value: 0x0000 0000 (0xFFFF FFFF)

LDO_ANA_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_RF_REG_BYPASS
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **RFD_RF_REG_BYPASS**: RF_REG Bypass mode.
0: Bypass mode disabled,
1: RF_REG in bypass mode

25.5.19 CBIAS0_ANA_ENG register (CBIAS0_ANA_ENG)

Address offset: 0x0174

Reset value: 0x0000 0088 (0xFFFF FFFF)

CBIAS0_ANA_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_CBIAS_IPTAT_TRIM[3:0]				RFD_CBIAS_IBIAS_TRIM[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **RFD_CBIAS_IPTAT_TRIM[3:0]**: overloaded value for cbias current trimming (when CBIAS0_TRIM_TST_SEL = 1)

Bits 3:0 **RFD_CBIAS_IBIAS_TRIM[3:0]**: overloaded value for cbias current trimming (when CBIAS0_TRIM_TST_SEL = 1)

25.5.20 CBIAS1_ANA_ENG register (CBIAS1_ANA_ENG)

Address offset: 0x0178

Reset value: 0x0000 0000 (0xFFFF FFFF)

CBIAS1_ANA_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBIAS0_TRIM_TST_SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.
								rw							

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CBIAS0_TRIM_TST_SEL**: When set, RFD_CBIAS_(IPTAT/IBIAS) _TRIM are used instead of HW trimmings

Bits 6:0 Reserved, must be kept at reset value.

25.5.21 SYNTHCAL0_DIG_OUT register (SYNTHCAL0_DIG_OUT)

Address offset: 0x0180

Reset value: 0x0000 0000 (0xFFFF FFFF)

SYNTHCAL0_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALAMP_OUT_6_0[6:0]						
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **VCO_CALAMP_OUT_6_0[6:0]**: VCO CALAMP value.

25.5.22 SYNTHCAL1_DIG_OUT register (SYNTHCAL1_DIG_OUT)

Address offset: 0x0184

Reset value: 0x0000 0001 (0xFFFF FFFF)

SYNTHCAL1_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALAMP_OUT_10_7[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **VCO_CALAMP_OUT_10_7[3:0]**: VCO CALAMP value

25.5.23 SYNTHCAL2_DIG_OUT register (SYNTHCAL2_DIG_OUT)

Address offset: 0x0188

Reset value: 0x0000 0040 (0xFFFF FFFF)

SYNTHCAL2_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALFREQ_OUT[6:0]						
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **VCO_CALFREQ_OUT[6:0]**: VCO CALFREQ value

25.5.24 SYNTHCAL3_DIG_OUT register (SYNTHCAL3_DIG_OUT)

Address offset: 0x018C

Reset value: 0x0000 0000 (0xFFFF FFFF)

SYNTHCAL3_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNTHCAL_DEBUG_BUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SYNTHCAL_DEBUG_BUS[7:0]**: Calibration debug bus.

Provide PLL calibration error details when SYNTHCAL0_DIG_ENG = 0xC:

- bit[7:4]: 0000
- bit3: CAL_ERROR
- bit2: CALAMP_ERROR
- bit1: CALFREQ_ERROR
- bit0: CALKVCO_ERROR

25.5.25 SYNTHCAL4_DIG_OUT register (SYNTHCAL4_DIG_OUT)

Address offset: 0x0190

Reset value: 0x0000 0018 (0xFFFF FFFF)

SYNTHCAL4_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOD_REF_DAC_WORD_OUT[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **MOD_REF_DAC_WORD_OUT[5:0]**: Calibration word

25.5.26 SYNTHCAL5_DIG_OUT register (SYNTHCAL5_DIG_OUT)

Address offset: 0x0194

Reset value: 0x0000 0007 (0xFFFF FFFF)

SYNTHCAL5_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBP_CALIB_WORD[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CBP_CALIB_WORD[3:0]**: CBP Calibration word

25.5.27 FSM_STATUS_DIG_OUT register (FSM_STATUS_DIG_OUT)

Address offset: 0x0198

Reset value: 0x0000 0000 (0xFFFF FFFF)

FSM_STATUS_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNTH_CAL_ERROR	Res.	Res.	STATUS[4:0]				
								r			r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SYNTH_CAL_ERROR**: PLL calibration error

Bits 6:5 Reserved, must be kept at reset value.

Bits 4:0 **STATUS[4:0]**: RF FSM state:

- 00000: IDLE
- 00001: ACTIVE1
- 00010: ENA_RF_REG
- 00011: ENA_CURR
- 00100: ACTIVE2
- 00101 to 01111: Not used
- 10000: ENA_TRANSFO_LDO
- 10001: SYNTH_SETUP
- 10010: CALIB10
- 10011: CALIB01
- 10100: CALIB11
- 10101: LOCKRXTX
- 10110: Not used
- 10111: Not used
- 11000: EN_RX
- 11001: EN_PA
- 11010: RX
- 11011Reserved
- 11100: TX
- 11101: Not used
- 11110: PA_DWN_ANA
- 11111: Not used

25.5.28 RSSI0_DIG_OUT register (RSSI0_DIG_OUT)

Address offset: 0x01A4

Reset value: 0x0000 0008 (0xFFFF FFFF)

RSSI0_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSI_MEAS_OUT_7_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RSSI_MEAS_OUT_7_0[7:0]**: Measure of the received signal strength.

25.5.29 RSSI1_DIG_OUT register (RSSI1_DIG_OUT)

Address offset: 0x01A8

Reset value: 0x0000 0008 (0xFFFF FFFF)

RSSI1_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSI_MEAS_OUT_15_8[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RSSI_MEAS_OUT_15_8[7:0]**: Measure of the received signal strength.

25.5.30 AGC_DIG_OUT register (AGC_DIG_OUT)

Address offset: 0x01AC

Reset value: 0x0000 0000 (0xFFFF FFFF)

AGC_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_ATT_OUT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AGC_ATT_OUT[3:0]**: AGC attenuation value

25.5.31 DEMOD_DIG_OUT register (DEMOD_DIG_OUT)

Address offset: 0x01B0

Reset value: 0x0000 0000 (0xFFFF FFFF)

DEMOD_DIG_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_END	PD_FOUND	AAC_FOUND	CI_FIELD[1:0]	
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RX_END**: rx_end

Bit 3 **PD_FOUND**: pd_found

Bit 2 **AAC_FOUND**: aac_found

Bits 1:0 **CI_FIELD[1:0]**: CI field

25.5.32 AGC2_ANA_TST register (AGC2_ANA_TST)

Address offset: 0x01BC

Reset value: 0x0000 0000 (0xFFFF FFFF)

AGC2_ANA_TST register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_ANTENNAE_USR_TRIM[2:0]			AGC2_ANA_TST_SEL
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:1 **AGC_ANTENNAE_USR_TRIM[2:0]**: the AGC antenna trimming value (when AGC2_ANA_TST_SEL = 1)

Bit 0 **AGC2_ANA_TST_SEL**: Selection.

0: default value is 0 (normal mode): the AGC antenna trimming value comes from the SoC integrating the MR_BLE IP

1: forced by this register (test mode): the AGC antenna trim value comes from the AGC2_ANA_TST[3:1] bit field value.

25.5.33 AGC0_DIG_ENG register (AGC0_DIG_ENG)

Address offset: 0x01C0

Reset value: 0x0000 004A (0xFFFF FFFF)

AGC0_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_ENABLE	AGC_THR_HIGH[5:0]					
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **AGC_ENABLE**: Enable AGC

Bits 5:0 **AGC_THR_HIGH[5:0]**: High AGC threshold

25.5.34 AGC1_DIG_ENG register (AGC1_DIG_ENG)

Address offset: 0x01C4

Reset value: 0x0000 0084 (0xFFFF FFFF)

AGC1_DIG_ENG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_LOCK_SYNC	AGC_AUTOLOCK	AGC_THR_LOW_6[5:0]					
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **AGC_LOCK_SYNC**: AGC locks when Access Address is detected (recommended)

Bit 6 **AGC_AUTOLOCK**: AGC locks when level is steady between high threshold and lock threshold

Bits 5:0 **AGC_THR_LOW_6[5:0]**: Low threshold for 6 dB steps

25.5.35 AGC10_DIG_ENG register (AGC10_DIG_ENG)

Address offset: 0x01E8

Reset value: 0x0000 0000 (0xFFFF FFFF)

Mapping for AGC step 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_0[1:0]		ATT_LNA_0	ATT_IF_0[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_0[1:0]**: attenuation at antenna level level for the AGC step 0
- 00: 0 dB (no attenuation)
- 01: -6 dB
- 10: -12 dB
- 11: -18 dB

Bit 3 **ATT_LNA_0**: Attenuation at LNA Level for the AGC step 0.
- 0: 0 dB (No attenuation)
- 1: -6 dB

Bits 2:0 **ATT_IF_0[2:0]**: attenuation at IF level for the AGC step 0
- 000: 0 dB (no attenuation)
- 001: -6 dB
- 010: -12 dB
- 011: -18 dB
- 100: -24 dB
- 101: -30 dB
- 11x: RFU, shall not be used

25.5.36 AGC11_DIG_ENG register (AGC11_DIG_ENG)

Address offset: 0x01EC

Reset value: 0x0000 0010 (0xFFFF FFFF)

Mapping for AGC step 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_1[1:0]		ATT_LNA_1	ATT_IF_1[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_1[1:0]**: attenuation at antenna level level for the AGC step 1

Bit 3 **ATT_LNA_1**: Attenuation at LNA Level for the AGC step 1

Bits 2:0 **ATT_IF_1[2:0]**: attenuation at IF level for the AGC step 1

25.5.37 AGC12_DIG_ENG register (AGC12_DIG_ENG)

Address offset: 0x01F0

Reset value: 0x0000 0020 (0x000F FFFF FFFF)

Mapping for AGC step 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_2[1:0]		ATT_LNA_2	ATT_IF_2[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_2[1:0]**: attenuation at antenna level for the AGC step 2

Bit 3 **ATT_LNA_2**: Attenuation at LNA Level for the AGC step 2

Bits 2:0 **ATT_IF_2[2:0]:** attenuation at IF level for the AGC step 2

25.5.38 AGC13_DIG_ENG register (AGC13_DIG_ENG)

Address offset: 0x01F4

Reset value: 0x0000 0030 (0xFFFF FFFF)

Mapping for AGC step 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_3[1:0]		ATT_LNA_3	ATT_IF_3[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_3[1:0]**: attenuation at antenna level for the AGC step 3

Bit 3 **ATT_LNA_3**: Attenuation at LNA Level for the AGC step 3

Bits 2:0 **ATT_IF_3[2:0]**: attenuation at IF level for the AGC step 3

25.5.39 AGC14_DIG_ENG register (AGC14_DIG_ENG)

Address offset: 0x01F8

Reset value: 0x0000 0038 (0xFFFF FFFF)

Mapping for AGC step 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_4[1:0]		ATT_LNA_4	ATT_IF_4[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_4[1:0]**: attenuation at antenna level for the AGC step 4

Bit 3 **ATT_LNA_4**: attenuation at LNA level for the AGC step 4

Bits 2:0 **ATT_IF_4[2:0]:** attenuation at IF level for the AGC step 4

25.5.40 AGC15_DIG_ENG register (AGC15_DIG_ENG)

Address offset: 0x01FC

Reset value: 0x0000 0039 (0xFFFF FFFF)

Mapping for AGC step 5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_5[1:0]		ATT_LNA_5	ATT_IF_5[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_5[1:0]**: attenuation at antenna level for the AGC step 5

Bit 3 **ATT_LNA_5**: Attenuation at LNA Level for the AGC step 5

Bits 2:0 **ATT_IF_5[2:0]**: attenuation at IF level for the AGC step 5

25.5.41 AGC16_DIG_ENG register (AGC16_DIG_ENG)

Address offset: 0x0200

Reset value: 0x0000 003A (0xFFFF FFFF)

Mapping for AGC step 6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_6[1:0]		ATT_LNA_6	ATT_IF_6[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_6[1:0]**: attenuation at antenna level for the AGC step 6

Bit 3 **ATT_LNA_6**: Attenuation at LNA Level for the AGC step 6

Bits 2:0 **ATT_IF_6[2:0]**: attenuation at IF level for the AGC step 6

25.5.42 AGC17_DIG_ENG register (AGC17_DIG_ENG)

Address offset: 0x0204

Reset value: 0x0000 003B (0xFFFF FFFF)

Mapping for AGC step 7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_7[1:0]		ATT_LNA_7	ATT_IF_7[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_7[1:0]**: attenuation at antenna level for the AGC step 7

Bit 3 **ATT_LNA_7**: Attenuation at LNA Level for the AGC step 7

Bits 2:0 **ATT_IF_7[2:0]**: attenuation at IF level for the AGC step 7

25.5.43 AGC18_DIG_ENG register (AGC18_DIG_ENG)

Address offset: 0x0208

Reset value: 0x0000 003C (0xFFFF FFFF)

Mapping for AGC step 8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_8[1:0]		ATT_LNA_8	ATT_IF_8[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_8[1:0]**: attenuation at antenna level for the AGC step 8

Bit 3 **ATT_LNA_8**: Attenuation at LNA Level for the AGC step 8

Bits 2:0 **ATT_IF_8[2:0]**: attenuation at IF level for the AGC step 8

25.5.44 AGC19_DIG_ENG register (AGC19_DIG_ENG)

Address offset: 0x020C

Reset value: 0x0000 003D (0xFFFF FFFF)

Mapping for AGC step 9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_9[1:0]		ATT_LNA_9	ATT_IF_9[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ATT_ANT_9[1:0]**: attenuation at antenna level for the AGC step 9

Bit 3 **ATT_LNA_9**: Attenuation at LNA Level for the AGC step 9

Bits 2:0 **ATT_IF_9[2:0]**: attenuation at IF level for the AGC step 9

25.5.45 RXADC_HW_TRIM_OUT register (RXADC_HW_TRIM_OUT)

Address offset: 0x0224

Reset value: 0x0000 001B (0xFFFF FFFF)

RXADC_HW_TRIM_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_RXADC_DELAYTRIM_Q[2:0]			HW_RXADC_DELAYTRIM_I[2:0]		
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:3 **HW_RXADC_DELAYTRIM_Q[2:0]**: control bits of the RX ADC loop delay for Q channel (provided by the HW trimming, automatically loaded on POR).

Bits 2:0 **HW_RXADC_DELAYTRIM_I[2:0]**: control bits of the RX ADC loop delay for I channel (provided by the HW trimming, automatically loaded on POR).

25.5.46 CBIAS0_HW_TRIM_OUT register (CBIAS0_HW_TRIM_OUT)

Address offset: 0x0228

Reset value: 0x0000 0088 (0xFFFF FFFF)

CBIAS0_HW_TRIM_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_CBIAS_IPTAT_TRIM[3:0]				HW_CBIAS_IBIAS_TRIM[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **HW_CBIAS_IPTAT_TRIM[3:0]**: CBIAS current (provided by the HW trimming, automatically loaded on POR).

Bits 3:0 **HW_CBIAS_IBIAS_TRIM[3:0]**: CBIAS current (provided by the HW trimming, automatically loaded on POR).

25.5.47 AGC_HW_TRIM_OUT register (AGC_HW_TRIM_OUT)

Address offset: 0x0230

Reset value: 0x0000 0006 (0xFFFF FFFF)

AGC_HW_TRIM_OUT register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_AGC_ANTENNAE_TRIM[2:0]			Res.
												r	r	r	

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:1 **HW_AGC_ANTENNAE_TRIM[2:0]**: AGC trim value (provided by the HW trimming, automatically loaded on POR).

Note: this value depends on the RF BOM on the board. Value provided by engineering is based on a dedicated BOM and must be overloaded by SW if the customer selects/defines another BOM.

Bit 0 Reserved, must be kept at reset value.

25.5.48 ANTSW0_DIG_USR register (ANTSW0_DIG_USR)

Address offset: 0x0240

Reset value: 0x0000 001C (0xFFFF FFFF)

ANTSW0_DIG_USR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TIME_TO_SAMPLE[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **RX_TIME_TO_SAMPLE[6:0]**: specifies the exact timing of the first I/Q sampling in the reference period.

Time unit is 250 ns.

Note: the value of this register is an offset to apply to a hard-coded 4.5 µs delay. The global delay (4.5 µs + programmable offset) is started from an internal trigger occurring before the Guard period on the air. The RX_TIME_TO_SAMPLE and RX_TIME_TO_SWITCH share the same internal trigger.

25.5.49 ANTSW1_DIG_USR register (ANTSW1_DIG_USR)

Address offset: 0x0244

Reset value: 0x0000 000B (0xFFFF FFFF)

ANTSW1_DIG_USR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TIME_TO_SWITCH[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RX_TIME_TO_SWITCH[5:0]**: specifies the exact timing of the antenna switching at receiver level (in AoA).

Time unit is 250 ns.

Note: the timing defined in this register is a delay from an internal trigger occurring before the Guard period on the air. The RX_TIME_TO_SAMPLE and RX_TIME_TO_SWITCH share the same internal trigger.

25.5.50 ANTSW2_DIG_USR register (ANTSW2_DIG_USR)

Address offset: 0x0248

Reset value: 0x0000 0029 (0xFFFF FFFF)

ANTSW2_DIG_USR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TX_TIME_TO_SWITCH[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TX_TIME_TO_SWITCH[6:0]**: specifies the exact timing of the antenna switching during transmission at LE_1M baud rate (in AoD).

Time unit is 125 ns.

Note: the timing defined in this register is a delay from an internal trigger occurring before the Guard period on the air (when transmit block starts sending the CTE to the modulator).

25.5.51 ANTSW3_DIG_USR register (ANTSW3_DIG_USR)

Address offset: 0x024C

Reset value: 0x0000 0023 (0xFFFF FFFF)

ANTSW3_DIG_USR register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TX_TIME_TO_SWITCH_2M[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TX_TIME_TO_SWITCH_2M[6:0]**: specifies the exact timing of the antenna switching during transmission at LE_2M baud rate (in AoD).

Time unit is 125 ns.

Note: the timing defined in this register is a delay from an internal trigger occurring before the Guard period on the air (when transmit block starts sending the CTE to the modulator).

The modulator latency differs between 1 M b/s and 2 Mb/s baud rate so 2 different delays need to be managed..

25.5.52 Trimming information

The Radio controller MR_BLE IP loads automatically HW trimming information located in the Flash memory of the SoC.

The trimmed information is:

- RX ADC delay for I and for Q channels
- IPTAT and BIAS current trimming for CBIAS block
- AGC trimming

Those trimming values are automatically loaded by the HW on reset and low power mode exit.

The loaded values are readable in dedicated radio registers (xx_HW_TRIM_OUT).

The AGC user trimming can be impacted by the BOM on the user board and may need to be overloaded. The SW can overload / replace the HW value

The AGC trimming consists in 1 information:

- AGC_ANTENNAE_TRIM_I
  - HW value readable in AGC_HW_TRIM_OUT[3:1] = HW_AGC_ANTENNAE_TRIM[2:0]
  - SW value writable in AGC2_ANA_TST[3:1] =.AGC_ANTENNAE_USR_TRIM[2:0]
  - SW overload feature activation through AGC2_ANA_TST[0] = AGC2_ANA_TST_SEL.

The HW values are reloaded on any reset and low power mode exit.

The SW values must be re-written after any reset or low power mode.

25.6 Radio FSM

The Radio FSM manages the analog radio block startup and stop sequences depending on requesting RF transfer.

25.6.1 Radio FSM sequence

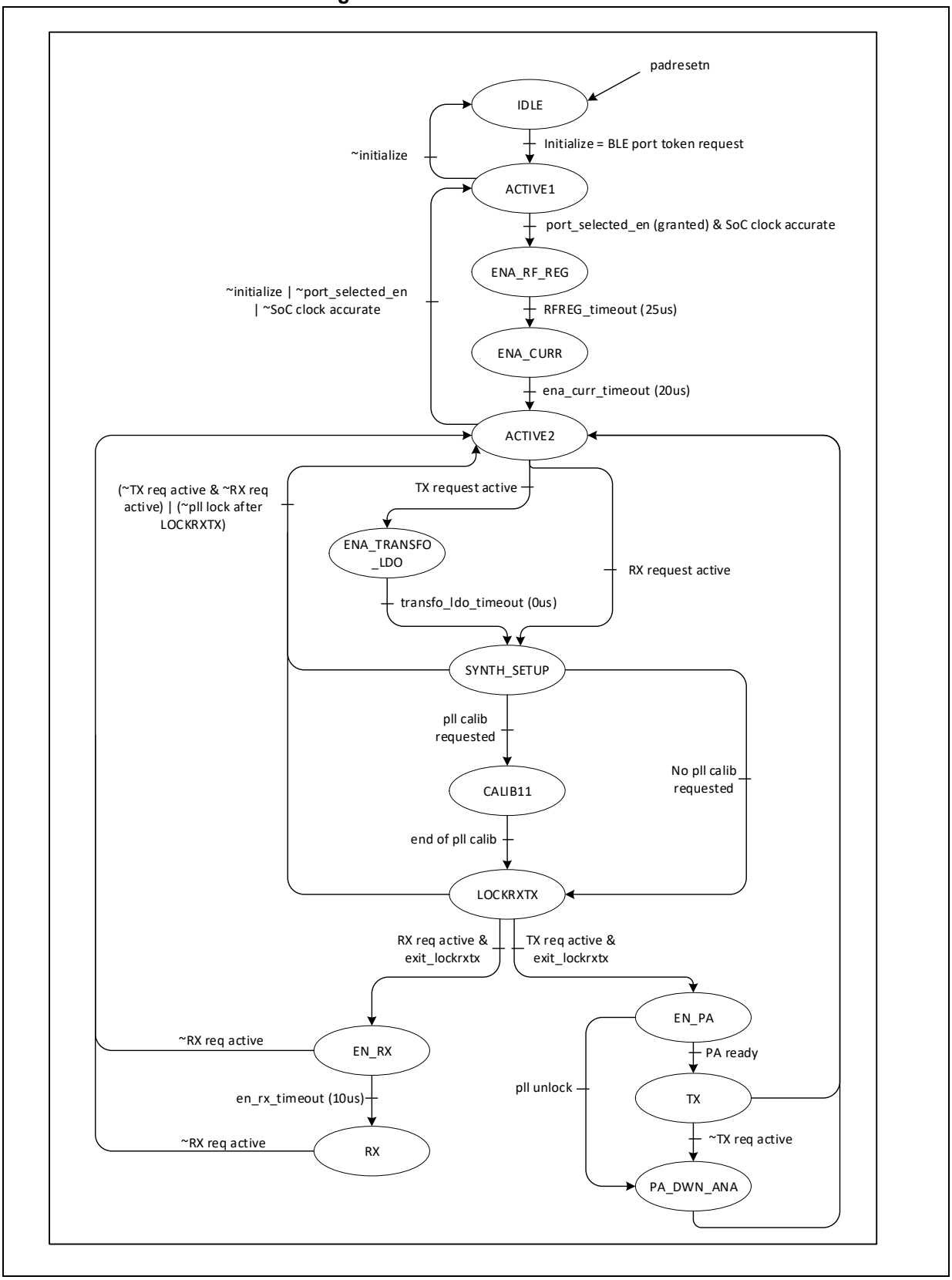
This paragraph lists the main steps in Radio FSM sequence.

- The Radio FSM stays in IDLE as long as the IP_BLE does not request the RRM token to indicate the radio is about to be used.
- Once the token is requested, the Radio FSM switches to ACTIVE1.
- When the device switches on the accurate fast clock (external XO) **AND** if the RRM semaphore granted one port (whatever the port), the Radio FSM goes to ACTIVE2 (through few intermediate steps to enable the RF LDO and central bias current).
- Once a RX or TX request is received, the Radio FSM switches to the RX or TX through intermediate steps to setup properly the analog.
- The Radio FSM goes back:
  - to ACTIVE2 as soon as RX or TX request is cleared and back to ACTIVE1 if the accurate clock is replaced by the dirty one
  - to IDLE if no more port requests a token to the RRM semaphore.

Figure 200: Radio FSM overview provides an overview of the Radio FSM states sequence.

Some transitions are triggered by hardware signals, but others are managed through timeout. The paragraph <XREF> lists the different timeouts.

Figure 200. Radio FSM overview



25.6.2 Radio FSM states overview

Table 127 lists for each state the exit condition and the duration in the state when there is a deterministic one.

Table 127. Radio FSM states summary (including exit conditions and timings)

Radio FSM State	State description	Condition to exit and duration in the state
IDLE	No RF activity requested: the analog RF IP is OFF.	The Bluetooth LE link layer requests a token to the RRM semaphore
ACTIVE1	Wait for port granted and SoC accurate clock indication.	Both HW conditions are fulfilled
ENA_RF_REG	Enable the RF LDO.	Timeout = 25 µs
ENA_CURR	Enable the reference current block inside the RF2G4.	Timeout = 20 µs
ACTIVE2	This state confirms all clock and power conditions are OK to accept a RF transfer request.	TX or RX request occurrence
ENA_TRANSFO_LDO	Enable the LDO for the Power Amplifier.	Timeout = 0 µs
SYNTH_SETUP	Start the RF PLL block.	No timeout. Exit immediately (1 MR_BLE clock cycle duration)
CALIB11	Start the PLL calibration block	End of calibration HW information.
LOCKRXTX **(*)**	Wait for RF PLL lock.	Timeout = – 40 µs when no PLL calibration – 80 µs when PLL calibration
EN_RX	Enable the analog RX chain.	Timeout = 10 µs
RX	Radio is in reception mode	End of RX event (RX timeout, RX done...)
EN_PA **(*)**	Start the power ramp-up sequence (8 steps) up to targeted power on the antenna	Ready signal informing targeted TX power is reached on the antenna (8 steps of 1.5 µs = 12 µs)
TX	Radio is in transmission mode	End of TX event (TX done or skipped)
PA_DWN_ANA	Disable the Power Amplifier block	Timeout = 5 µs

(*): ***The Radio FSM may abort the sequence and return to ACTIVE2 from those two states depending on RF PLL lock information:***

- LOCKRXTX: the decision occurs at the very end of the state to decide if the sequence should go on (PLL locked) or abort (PLL lock failed)
- EN_PA: the decision to abort can occur at any time inside this state as soon as the RF PLL is no more locked (PLL unlocked)

If the PLL unlock event occurs outside those two states, the Radio FSM does not interfere. The SW is informed through an interrupt and is in charge to manage the situation.

The current state information is available in the FSM_STATUS_DIG_OUT radio register accessible by direct APB access (see section *Section 25.5: Radio registers*).

The Radio FSM uses timeout to exit states linked to analog block settlement to guarantee a deterministic duration between ACTIVE2 and TX or RX state at any occurrence. this is mandatory to be able to fit with Bluetooth protocol timings requirements in peer-to-peer communication flow.

Those deterministic durations are presented in *Table 128*.

Table 128. ACTIVE2 to TX or RX state duration

Sequence	Duration	Comments
ACTIVE2 → TX with RF PLL calibration	92 µs	To be used when the TX is done on a new frequency (channel) versus previous RF transfer
ACTIVE2 → TX without RF PLL calibration	52 µs	To be used when the TX is done on the same frequency (channel) as previous RF transfer
ACTIVE2 → RX with RF PLL calibration	90 µs	To be used when the RX is done on a new frequency (channel) versus previous RF transfer
ACTIVE2 → RX without RF PLL calibration	50 µs	To be used when the RX is done on the same frequency (channel) as previous RF transfer

25.6.3 Radio FSM interrupts

The Radio FSM provides a dedicated interrupt output signal to the system.

The Radio FSM generates 6 individually maskable interrupts grouped in a RfFsm_event[5:0] list:

- RfFsm_event[0] = PLL Lock timeout
  - set when the counter to exit LOCKRXTX expires
  - whatever the lock status (PLL locked or not locked at timer expiration)
- RfFsm_event[3] = PLL unlock detection
  - set when the PLL lock signal falls after the lock detection step.
- RfFsm_event[4] = PLL lock failed
  - set if the PLL lock is not high when the Radio FSM exits the LOCKRXTX state
  - or set if the PLL is no more locked during EN_PA state or on exit of EN_PA state
- RfFsm_event[5] = PLL calibration error
  - set if a PLL calibration error occurs during PLL calibration step.
  - Problem can concern the amplitude calibration and/or the frequency calibration and/or the KVCO calibration.
  - In this case, the detail can be read in SYNTH3_DIG_OUT[3:0] if SYNTHCAL0_DIG_ENG[3:0] = 0xC:

    SYNTHCAL3_DIG_OUT[3] = CAL_ERROR

    SYNTHCAL3_DIG_OUT[2] = CALAMP_ERROR

    SYNTHCAL3_DIG_OUT[1] = CALFREQ_ERROR

    SYNTHCAL3_DIG_OUT[0] = CALKVCO_ERROR

RfFsm_event[2] and RfFsm_event[1] are reserved.

All the sources are combined into a single signal to be connected outside the Radio controller MR_BLE IP to the interrupt controller of the SoC (see *Table 122: Radio controller MR_BLE interruptions summary* for mapping in the STM32WB09xE).

The interrupts can be enabled/disabled individually through the Radio controller APB registers:

- Enable/disable through RADIO_CONTROL_IRQ_ENABLE register
- Reading the RADIO_CONTROL_IRQ_STATUS register returns the interrupts status.
- Writing a ‘1’ in the RADIO_CONTROL_IRQ_STATUS[x] clear the associated interrupt flag.

See *Section 25.8: Radio controller registers* for more details.

25.7 Radio controller

The radio controller is a small block in charge of two features:

- Slow clock period measurement,
- Radio FSM interrupt management.

25.7.1 Slow clock measurement

The radio controller contains a block dedicated to the slow clock measurement.

This measurement:

- is launched automatically by the hardware when the system clock switches on accurate clock (external XO).
- can be launched by the software when needed (by writing zero in CLK32K_PERIOD register)

The result provided by this block is both a period and a frequency information (in two separate results registers).

The software can program the window of measurement (in slow clock cycle number) and period result is provided in 16 MHz half-period unit.

25.7.2 Radio FSM interrupt management

During the sequences, the Radio FSM generates some interruptions to monitor some unexpected behavior at analog level. As the Radio FSM block does not have any APB interface, the interrupt control and status flags are managed inside the Radio controller block through APB registers:

- RADIO_CONTROL_IRQ_ENABLE register to enable the wanted interrupt sources.
- RADIO_CONTROL_IRQ_STATUS register to get the status (on read) and to clear the interrupt (by writing ‘1’ on the associated bit)

See Radio FSM interrupts *Section 25.6.3: Radio FSM interrupts* for more details on interrupt root causes.

25.8 Radio controller registers

The **RADIO_CONTROL_BLOCKBaseAddress** keyword used for all the register base address information corresponds to the Radio controller registers base address decided by the SoC when integrating the IP.

*Note: **RADIO_CONTROL_BLOCKBaseAddress** is 0x6000_1000 in the STM32WB09xE product.*

Table 129. Radio control register list

Offset	Register Name	Register Title	Reset
0x0000	RADIO_CONTROL_ID	RADIO_CONTROL_ID register	0x0000 3000
0x0004	CLK32COUNT_REG	CLK32COUNT_REG register	0x0000 0017
0x0008	CLK32PERIOD_REG	CLK32PERIOD_REG register	0x0000 0000
0x000C	CLK32FREQUENCY_REG	CLK32FREQUENCY_REG register	0x0000 0000
0x0010	RADIO_CONTROL_IRQ_STATUS	RADIO_CONTROL_IRQ_STATUS register	0x0000 0000
0x0014	RADIO_CONTROL_IRQ_ENABLE	RADIO_CONTROL_IRQ_ENABLE register	0x0000 0000

25.8.1 RADIO_CONTROL_ID register (RADIO_CONTROL_ID)

Address offset: 0x0000

Reset value: 0x0000 3000 (0xFFFF FFFF)

Identification register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT[3:0]				VERSION[3:0]				REVISION[3:0]				Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **PRODUCT[3:0]**: incremented on major features add-on like new Bluetooth LE SIG version support
- 0x01: MR_BLE_V1
- 0x02: MR_BLE_V2
- 0x03: MR_BLE_V3

Bits 11:8 **VERSION[3:0]**: Cut Number

Bits 7:4 **REVISION[3:0]**: Incremented for metal fix version

Bits 3:0 Reserved, must be kept at reset value.

25.8.2 CLK32COUNT_REG register (CLK32COUNT_REG)

Address offset: 0x0004

Reset value: 0x0000 0017 (0xFFFF FFFF)

Window length register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_COUNT[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **SLOW_COUNT[8:0]**: program the window length (in slow clock period) for slow clock measurement.

Slow clock is measured in a window of SLOW_COUNT+1 slow clock cycles

Note:

- when programming 0xFF, the window is 256 slow clock cycles
- to have a good precision, not less than 0x17 is recommended

25.8.3 CLK32PERIOD_REG register (CLK32PERIOD_REG)

Address offset: 0x0008

Reset value: 0x0000 0000 (0xFFFF FFFF)

slow clock period

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_PERIOD[18:16]		
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLOW_PERIOD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **SLOW_PERIOD[18:0]**: indicates slow clock period information. The result provided in this field corresponds to the length of SLOW_COUNT periods of the slow clock (32kHz) measured in 16 MHz half-period unit.

Example:

if SLOW_COUNT=0x17=23d and SLOW_PERIOD=24000d

Slow clock period = SLOW_PERIOD / (16e6 x 2 x (SLOW_COUNT+1)) = 24000 / (16e6 x 2 x 24) = 31.25e-6

A new calculation can be launched by writing zero in CLK32PERIOD register. In this case, the time window uses the value programmed in SLOW_COUNT field.

25.8.4 CLK32FREQUENCY_REG register (CLK32FREQUENCY_REG)

Address offset: 0x000C

Reset value: 0x0000 0000 (0xFFFF FFFF)

slow frequency register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SLOW_FREQUENCY[26:16]										
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLOW_FREQUENCY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:0 **SLOW_FREQUENCY[26:0]**: value equal to (2^39/ SLOW_PERIOD).

25.8.5 RADIO_CONTROL_IRQ_STATUS register (RADIO_CONTROL_IRQ_STATUS)

Address offset: 0x0010

Reset value: 0x0000 0000 (0xFFFF FFFF)

interrupt status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RADIO_FSM_IRQ[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_CLK_IRQ
		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1								rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **RADIO_FSM_IRQ[5:0]**: Radio FSM interrupt status (also called RfFsm_event_irq).
0: no pending interrupt
1: pending interrupt
RfFsm_event [5] = PLL calibration error
RfFsm_event [4] = PLL lock failed
RfFsm_event [3] = PLL unlock detection
RfFsm_event [2] = reserved
RfFsm_event [1] = reserved
RfFsm_event [0] = lock_timeout
Write '1' to clear the interrupt

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SLOW_CLK_IRQ**: slow clock measurement end of calculation interrupt status
When read:
0: no pending interrupt,
1: pending interrupt: Slow clock period/frequency values are available.
Write '1' to clear the interrupt
Note: those flags are set regardless associated RADIO_CONTROL_IRQ_ENABLE mask (which is used onyl to raise or not the interrupt request towards the CPU)

25.8.6 RADIO_CONTROL_IRQ_ENABLE register (RADIO_CONTROL_IRQ_ENABLE)

Address offset: 0x0014

Reset value: 0x0000 0000 (0xFFFF FFFF)

interrupt enable register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RADIO_FSM_IRQ_MASK[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_CLK_IRQ_MASK
		rw	rw	rw	rw	rw	rw								rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **RADIO_FSM_IRQ_MASK[5:0]**: mask for each RfFsm_event (Radio FSM) interrupt.
0: Interrupt disabled
1: Interrupt enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SLOW_CLK_IRQ_MASK**: mask slow clock measurement interrupt
0: Interrupt disabled
1: Interrupt enabled.

25.9 IP_BLE

The Bluetooth LE link layer of the IP_BLE is a programmable automate which can act as a central or a peripheral node.

The Bluetooth LE link layer embeds a sequencer which automatically reads data and job request from link tables and link lists prepared in advance by the CPU in retention RAM. This allows the Bluetooth LE link layer starting a Bluetooth LE reception or transmission directly at low power mode exit while the CPU is still booting and not yet able to manage any firmware action.

25.9.1 Overview

The IP_BLE embeds:

- a Bluetooth Link layer,
- a modulator,
- a demodulator

The Bluetooth LE link layer manages:

- the reception and transmission sequences including channel hopping,
- on-the-fly encryption thanks to an embedded AES (which can also be used by the SW to compute LE Privacy and manual encryption),
- the antenna switching feature

The Bluetooth LE link layer embeds a sequencer which uses information located in RAM tables to manage the RF transfer and part of the Bluetooth LE protocol. Those RAM tables need to be prepared by the SW.

In general, the process to generate a Bluetooth LE transfer is the following:

- the CPU prepares some tables in RAM containing information about next Bluetooth LE transfer(s)
- the CPU programs one of the 3 possible timers that triggers/starts a sequence
- When the selected timer matches the programmed value, the sequencer starts to execute a Bluetooth LE sequence. This implies:
  - to read the RAM table to know what to do,
  - to (re-)program the radio register if needed,
  - to launch the Radio FSM with a Rx or Tx request depending on requested transfer.
    - once the analog RF is ready to transmit (respectively receive), to read the data payload from RAM (respectively to write the data payload in RAM)
- Once the sequence is done (successfully or with errors), the sequencer writes back in RAM updated information (flags, pointers, etc.)
- Once RAM write back is over, the sequencer sends an interrupt to the CPU, related to interrupt mask programmed by the CPU (through the RAM table)

25.9.2 Bluetooth LE link layer sequencer

The sequencer needs a trigger event to start any action.

Then the sequencer manages a transmission or reception (or no) sequence depending on the RAM table content it reads.

Possible trigger timers for the sequencer

There are 3 different timers that can trig a Bluetooth LE sequence:

1. Wakeup timer:
   - this timer is based on absolute time (slow clock granularity),
   - this timer is inside the Wakeup block and is the only one able to wake up the IP_BLE (and the SoC) from a low power state
   - this timer is enabled by setting the BLUE_SLEEP_REQUEST_MODE[30] = BLE_WAKEUP_EN bit and the trigger event time is programmed in the BLUE_WAKEUP_TIME[31:0]
   - If a RRM command is enabled (through the StatMach table, RadioComListEna field), the sequencer requests the Command0 to the RRM-UDRA block during the sequence.
2. Timer1 timer
   - this timer is managed by the IP_BLE sequencer itself but using the interpolated time (see section *Interpolated time*),
   - the Timer1 is in fact a comparator between interpolated absolute time provided by the Wakeup block and a match value located in the sequencer,
   - the granularity of this timer is 16 x slow clock period,
   - this timer is enabled by setting DESTINATION[1:0] = “10” in the TIMEOUTDESTREG APB register in IP_BLE,
   - the Timer1 trigs an event when the current interpolated time matches (or has gone past) the scheduled time programmed in TIMEOUTREG APB register,
   - this timer cannot be used in low power mode,
   - If a RRM command is enabled (through the StatMach table, RadioComListEna field), the sequencer requests the Command1 to the RRM-UDRA block during the sequence.
3. Timer2 timer
   - this timer is based on a relative time and starts counting at the end of the previous transmission/reception,
   - the granularity of this counter is 1 µs,
   - this timer is located inside the sequencer and cannot be used in low power mode,
   - Timer2 triggers an event when the counter matches the value programmed in the Timer2[19:0] bit field of the TxRxPack RAM table,
   - this timer is enabled by setting the Timer2En bit in the TxRxPack RAM table of a sequence, allowing a trigger event for the next sequence (TX/RX sequence)
   - this timer is supposed to be used for short time delay between two Bluetooth transfers (to manage T_IFS or T_MSS for instance),
   - If a RRM command is enabled (through the StatMach table, RadioComListEna field), the sequencer requests the Command2 to the RRM-UDRA block during the sequence.

How to manage these 3 timers:

The 3 timers are managed (enabled/disabled) different way. The following paragraph explains how to manage according to the use-case.

Each timer is one-shot. This means once it expires, it stops and the software has to reprogram/re-enable it for a new sequence.

Furthermore, as there is no mechanism to prevent more than one timer to be active at the same time, the software must ensure it does not start a timer while another one is already on-going for the next sequence.

Here is a summary of the enable/disable method for each timer:

- the Wakeup timer is enabled and programmed through the Wakeup BLUE_WAKEUP_TIME APB register and has no interference with the two others so may be enabled in addition of Timer1 or Timer2.
- the Timer1:
  - duration is programmed only through the IP_BLE TIMEOUTDEST APB register,
  - enable is done only through the IP_BLE TIMEOUTDESTREG APB register,
  - disable can be done through the IP_BLE TIMEOUTDESTREG APB register
  - a HW disable is automatically done by the sequencer when it treats the Timer2 enable information in the TxRxPack RAM table under execution (whatever the TxRxPack.Timer2En bit value).
- the Timer2:
  - duration can be programmed either through the IP_BLE TIMEOUTDESTREG APB register or through the TxRxPack table,
  - can be enabled/disabled either through the IP_BLE TIMEOUTDESTREG APB register or through the TxRxPack table.

Note:

- Programming respectively Timer1 or Timer2 through the IP_BLE TIMEOUTDESTREG APB register automatically disables the Timer2 or Timer1 respectively.
- If the Bluetooth LE sequence ends on a receive timeout (STATUSREG.RCVTIMEOUT = 1), the Timer2 is not started even if the TxRxPack.Timer2En associated to this reception was 1.
- *Even if the Timer2 can be enabled through the IP_BLE TIMEOUTDESTREG APB register, it is recommended not to do it in application flow and to use RAM table.*
- TIMEOUTDESTREG[1:0] and TIMEOUTREG[31:0] need to be programmed at least 15 microseconds before the required start trigger

Bluetooth LE sequence description

As explained before, the Bluetooth LE sequence starts on the trigger event (from the wakeup timer or the Timer1 or the Timer2).

The sequence is managed in three mains phases:

- Initialization (split in 3 sub-steps)
- reception or transmission
- Context saving (to write back some updated field in RAM tables)

At the end of those phases, an interrupt (if at least one active source enabled) is generated to the CPU.

Note: The status flags and potential associated interrupt are set only at the end of the sequence and not in real time when the event that generates them occurs.

The STATUSREREG.SEQDONE flag (and INTERRUPT1REG.SEQDONE if enabled in the GlobalStatMach table) is always raised when the sequencer exits a sequence started by a trig event, whatever the process it followed (exit on error at any steps or run up to the end without problems). For this reason, this specific flag is not mentioned/repeated each time in this-chapter.

The first RAM access done by the sequencer on any trigger event is to get the GlobalStatMach word 0x01 to check the Active bit to know if the parameters the sequencer is about to read in the RAM table for the 1st INIT phase can be considered as ready and up to date.

If the Active bit is low, the sequence stops, and the only actions done are:

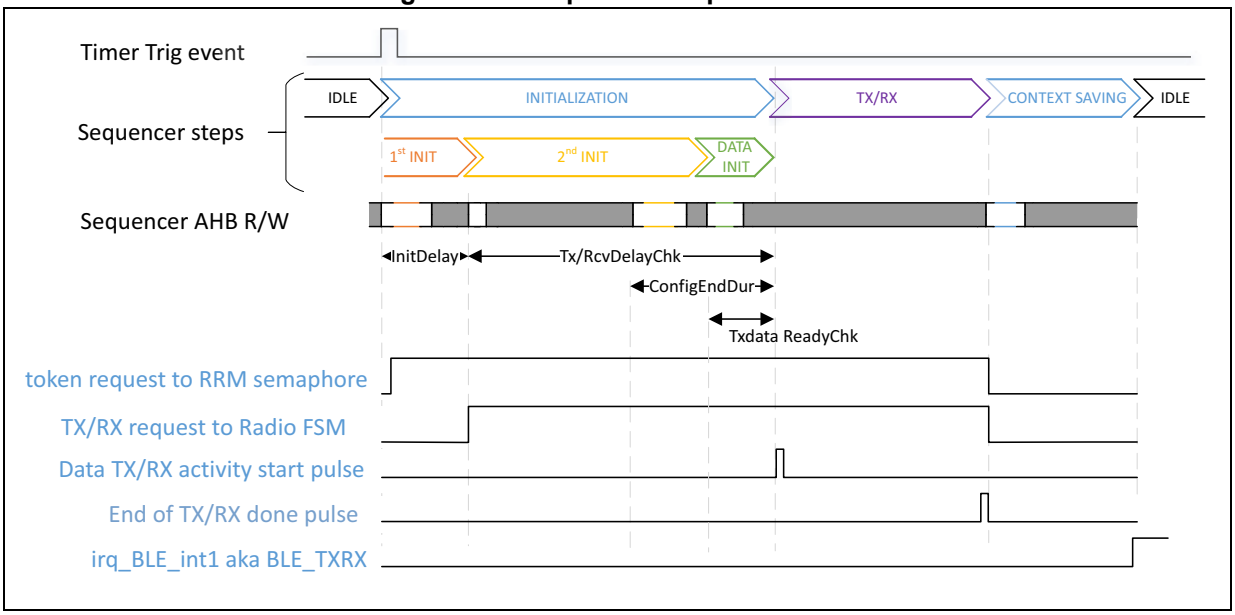
- setting NOACTIVELERROR flag in the STATUSREG IP_BLE APB register
- and if IntNoActiveLError is set in the GlobalStatMach, setting the NOACTIVELERROR in INTERRUPT1REG IP_BLE APB register and generating an associated interrupt.

If the GlobalStatMach.Active bit is set, then the sequencer starts the initialization phase.

Note: An automatic Active bit auto clear during context saving phase of the sequence can be enabled through the ChkFlagAutoClearEna bit in the GlobalStatMach. This avoids unexpected TX/RX due to sequencer trigger event while the SW did not update the RAM tables (kind of SW acknowledge to allow next transfer).

Table 201 provides an overview of the sequencer steps and control versus other blocks.

Figure 201. Sequencer steps overview



First initialization phase

During the first initialization phase, the sequencer only reads the minimum information it needs in the RAM table to be able to start the Radio FSM for a reception or a transmission at the end of this phase.

The sequencer launches up to 3 parallel tasks:

1. Set (or maintain if KeepSemaReq bit was set in the TxRxPack RAM table of the previous sequence) the take_req signal toward the RRM Semaphore block to request/keep the token

to access to the radio resources.

2. If the RadioComListEna bit is set in the current StatMach table, send a command to the RRM UDRA:
   – Command 0 if trig event is the Wakeup timer,
   – Command 1 if trig event is the Timer1,
   – Command 2 if the trig event is the Timer2.
3. Get the minimum information needed to be able to start the Radio FSM (transmission or reception, channel number, PLL calibration requested or not, etc.)

At the end, this task also computes the channel number through the channel incrementer hardware block if requested and writes few radio registers (MOD_DEM_DIG_USR, RADIO_FSM_USR and PHYCTRL_DIG_USR) according to information from the RAM tables.

This first initialization step ends on a timeout defined by a bit field in the GlobalStatMach:

- WakeupInitDelay (time unit is 16 x slow clock so typically 512kHz) when trig event source is the wakeup timer,

Note: despite the wakeup trigger event to start the sequence is based on a slow clock granularity, typ. 32kHz (as the trigger occurs at BLUE_WAKEUP_TIME[31:4]), the sequencer waits until interpolate time is BLUE_WAKEUP_TIME[31:0 + WakeUpInitDelay to exit the 1st INIT step which means the 512kHz granularity is respected.

- Timer12InitDelayCal (time unit is 1 µs) when the trig event is the Timer1 or when the trig event is the Timer2 and CalReq bit in TxRxPack table is set (PLL calibration requested),
- Timer2InitDelayNoCal (time unit is 1 µs) when the trig event is the Timer2 and CalReq bit in the TxRxPack table is low (no PLL calibration requested).

InitDelay is used as a generic name for this duration to simplify the documentation as it can be 3 different bit fields that defines it depending on the configuration.

Note: The main constraint on this delay depends on the previous setup:

- If KeepSemaReq bit was set in the TxRxPack of the previous transfer, then the Radio FSM stayed in ACTIVE2 and the main constraint to define the delay is the AHB accesses to read the RAM tables.
- If KeepSemaReq bit was low in the TxRxPack of the previous transfer, then the RRM Semaphore released the token and Radio FSM went back to IDLE. Then the main constraint for the delay is the duration for the Radio FSM to reach ACTIVE2 state from IDLE (25 µs for ENA_RF_REG step and 20 µs for ENA_CURR step).
- In parallel, if accurate clock was turned off, there is also the delay to have accurate clock available to consider.

Caution: Whatever the trig event source, this **InitDelay** management in the sequencer uses the system clock and the user must ensure that the system runs on an accurate clock to have a precise delay.

When the ***InitDelay*** timeout expires, the sequencer checks several conditions to decide if it switches to the second initialization step or exits with error. The checked conditions are:

- Radio FSM reached ACTIVE2 state meaning it is ready to receive a TX or RX request (Üand system clock is the accurate clock),
- All RAM accesses and radio register writings to be done by the sequencer during the first initialization step are over.
- No configuration error occurred (see *Configuration error* for more details)

If all the conditions are true, then the sequencer:

- sends a TX or RX request to the Radio FSM depending on transfer direction indicated by TxMode bit in the current StatMach table,
- and switches to the second initialization step.

If at least one of the conditions is false then:

- the sequence rises the flag(s) associated to the error(s). It can be:
  - STATUSREG.CONFIGERROR bit if a configuration error has been detected,
  - STATUSREG.ACTIVE2ERROR bit if the Radio FSM is not in ACTIVE2 at the end of the ***InitDelay***,
  - STATUSREG.SEMATIMEOUTERROR bit if the RRM semaphore did not grant the IP_BLE on time,
- No RAM write back action is done.
- The error bits set in STATUSREG register also appears in INTERRUPT1REG if their associated interrupt enable bit is set in the GlobalStatMach table.

Second initialization step

The 2nd INIT step is used by the sequencer to get all the information from the RAM tables linked to the RF transfer to proceed (except DataPtr and TxDataReady bit fields).

This means the software must have filled all the RAM tables information (except DataPtr and TxDataReady bit fields) when the ***InitDelay*** timeout expires.

The 2nd INIT step starts when the TX or RX request is sent to the Radio FSM. the first action of the sequencer is to read few delays in the GlobalStatMach. These delays are needed during the 2nd INIT and DATA INIT steps.

This 2nd INIT step ends on a timeout based on 2 information read in the GlobalStatMach:

1. init_radio_delay (in µs), used as a generic name for this duration to simplify the documentation: it is one of possibility out of 4 different bit fields depending on the transfer configuration:
   - TransmitNoCalDelayChk when the transfer is a TX and no PLL calibration is requested (CalReq bit is low),
   - TransmitCalDelayChk when the transfer is a TX and a PLL calibration is requested (CalReq bit is set),
   - ReceiveNoCalDelayChk when the transfer is a RX and no PLL calibration is requested (CalReq bit is low),
   - ReceiveCalDelayChk when the transfer is a RX and a PLL calibration is requested (CalReq bit is set),
2. TxdataReadyCheck: delay given to the sequencer to get the two last information which are DataPtr and TxDataReady information in the RAM table. These last readings are done in the third initialization step called DATA_INIT.

The 2nd INIT ends after “init_radio_delay – TxdataReadyCheck” µs.

From the 2nd INIT step, the Radio FSM is running in parallel of the sequencer getting information in the RAM tables. The user must ensure the init_radio_delay duration does not exceed the RF analog setup time up to power on the antenna for a transmission (or ready to receive on the antenna). This means the 2nd INIT step must not exceed:

- the duration of the Radio FSM to go from ACTIVE2 to TX state for a transmission with few µs of margin,
- the duration of the Radio FSM to go from ACTIVE2 to RX state for a reception.

Note: For transmission, the init_radio_delay timeout must expire before the Radio FSM is in TX mode to avoid missing the start of the preamble sending on the antenna (othrwise garbage is sent while preamble is supposed to be output).

For a reception, the init_radio_delay must expire close to the switch in RX state of the Radio FSM, knowing the RcvTimeout counter starts when the init_radio_delay expi

At very beginning of the 2nd INIT step:

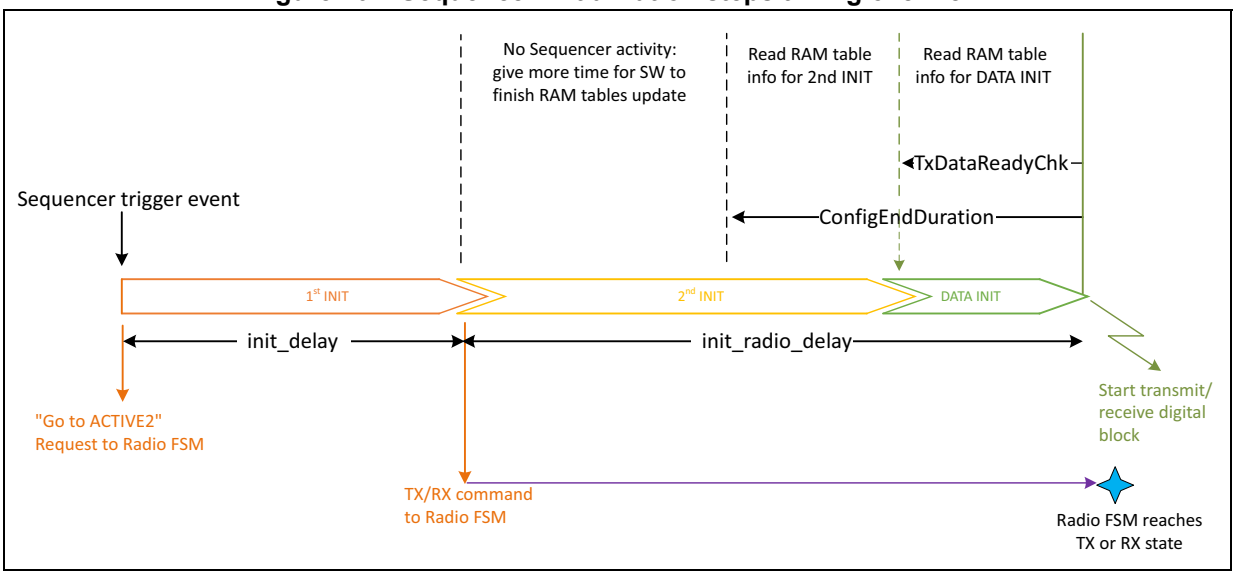
- the sequencer starts an internal relative timer.
- In parallel, the sequencer reads the init_radio_delay, ConfigEndDuration and TxdataReadyCheck information in the GlobalStatMach.

The 2nd INIT step really starts to fetch information related to the transfer in the RAM tables when the relative timer reaches “init_radio_delay – ConfigEndDuration”.

The GlobalStatMach.ConfigEndDuration bit field allows delaying the reading of the transfer information contained in the RAM tables by the sequencer. The goal of this delay is to provide more margin to the SW to fill the RAM table information that is read during 2nd INIT. This is possible as the RAM table read session is shorter than the analog radio setup duration.

Figure 126 provides a summary of the timing information contributors for the initialization steps.

Figure 202. Sequencer Initialization steps timing overview



Data initialization step

This Data INIT step starts when the 2nd INIT step ends.

During this step, the sequencer only gets 2 values from the table:

- TxDataReady bit in the TxRxPack indicating enough bytes are present in the TX payload data buffer (in case of transmission only),
- DataPtr bit field in the TxRxPack.

The GlobalStatMach.TxdataReadyCheck is used to delay the start of this DATA INIT step to let more time to the software to provide the data pointer (and first values to transmit if transfer is a transmission).

The DATA INIT step ends when the relative timer (started at the beginning of the 2nd INIT step) reaches init_radio_delay:

- If all conditions are OK (AllTableReady read at 1, TxDataReady read at 1 for a transmission, Radio FSM still in a state between ACTIVE2 and RX or TX, command_end received from the RRM if an UDRA command was launched in parallel), a start pulse is sent to the receive/transmit block for a reception/transmission.
- Else no start pulse is sent to the receive/transmit block and status/interrupt flags are updated in the IP_BLE APB registers (no RAM write back occurs).

For transmission, a synchronization mechanism is in place between the transmit block and the Radio FSM: the transmit block waits for the TX state information from the Radio FSM to know when data can be sent to the modulator. As the transmit block is supposed to receive the start pulse from the sequencer before the Radio FSM reaches the TX state, a wait window is defined to avoid waiting forever: this time window is defined in the GlobalStatMach.TxReadyTimeout bit field.

Caution: It is the responsibility of the software to ensure that the init_radio_delay, the ConfigEndDuration and the TxdataReadyCheck values are coherent to guarantee both data ready on time in the table and start pulse sent on time to the receive/transmit block.

Transmission / reception step

The transmission / reception step starts when the start pulse is sent by the sequencer to the transmit or to the receive block.

This step ends when the transmit/receive block indicates the transfer is done:

- all data transmitted for a transmission,
- a frame has been received or the programmed timeout to wait for a reception expired without any reception.

Important:

- When a transmission is completed, the Timer2, if enabled, starts counting only when GlobalStatMach.TxdelayEnd is elapsed
- When a reception is completed, if the exit reason is a timeout, the Timer2 does not start

Context saving step

The context saving steps consists in RAM write back operation in some RAM table words to update with the result of the RF transfer that just ended.

This step starts when the sequencer gets the transfer done information from the transmit or receive block.

The RAM write back impacts the following RAM table elements:

- GlobalStatMach Word1 if the GlobalStatMach.ChkFlagAutoClearEna bit is set:
  - clear the Active bit
  - Write back the rest of the bit field of this word1 with value previously read by the sequencer in the RAM table.
- StatMach Word0: update SN, NESN, remapped channel and next transfer direction (TxMode)
- StatMach Word1: update the TxPoint[31:0] with TxPointNext[31:0] or keep the same (see *Pointer management and packet counter* for pointer management details).
- StatMach Word2: update the RcvPoint[31:0] with RcvPointNext[31:0] or keep the same (see *Pointer management and packet counter* for pointer management details).
- StatMach Word3: update the TxPointPrev[31:0] with TxPoint[31:0] or keep the same (see *Pointer management and packet counter* for pointer management details).
- StatMach Word4: update the RcvPointPrev[31:0] with RcvPoint[31:0] or keep the same (see *Pointer management and packet counter* for pointer management details).
- StatMach Word5: update the TxPointNext[31:0] or keep the same (see *Pointer management and packet counter* for pointer management details).
- StatMach Word6: update the PCntTx[31:0] or keep the same (see *Pointer management and packet counter* for pointer management details).
- StatMach Word7: update the PCntTx[39:32] and PCntRcv[23:16] or keep the same (see *Pointer management and packet counter* for pointer management details).
- StatMach Word8:
- update the PCntRcv[39:24] or keep the same (see *Pointer management and packet counter* for pointer management details),
- rest of the Word8 is written back with value previously read by the sequencer in the RAM table

Bluetooth LE sequence summary

The sequences of operations characterizing a transmission and a reception are summarized in the following timing diagrams.

Figure 203. TX sequence

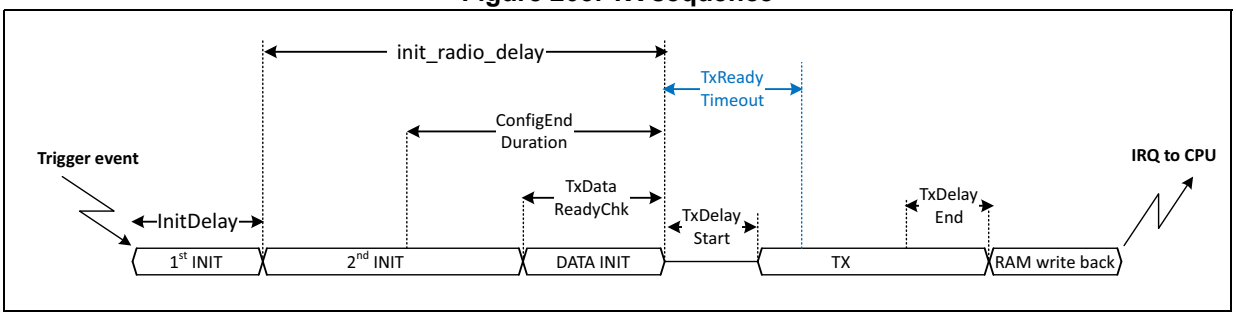
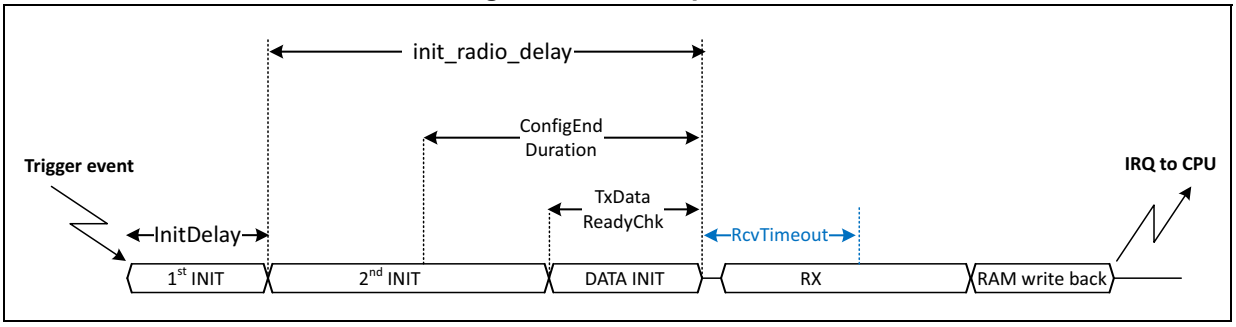


Figure 204. RX sequence



Possible root causes of aborted sequence

It may happen that the sequence is not started or interrupted before the end for different reasons.

In any case, the SEQDONE flag (and interrupt if enabled) occurs at the end of a sequence whatever the status (successful or failed).

Then some status/error flags (with associated maskable interrupt) are available to get the reason of abortion or to have complementary information on the sequence that just occurred.

Active bit is low

The sequence is stopped just after the trigger event because the Active bit in the GlobalStatMach RAM table is read equal to 0.

If active bit is low, then:

- no sequence is started (Radio FSM and transmit/receive blocks not informed),
- no timer management is done (no update / reprogramming of the different timers is done),
- no RAM write back occurs, RAM tables are left unchanged,
- the NOACTIVELERROR bit is set in the STATUSREG IP_BLE APB register is set,
- if the IntNoActiveLError bit in the GlobalStatMach word 0x05 is set, the NOACTIVELERROR bit is set in the INTERRUPT1REG IP_BLE APB registers and a IP_BLE interrupt is generated to the CPU
  - The enable mask is readable in INTERRUPT1ENABLEREG.NOACTIVELERROR bit.

RRM Semaphore does not grant the IP_BLE on time

On trigger event and if active bit is high, the sequencer requests the token to the RRM Semaphore to have the control of the radio resources.

If at the end of the initialization delay (WakeupInitDelay or Timer12InitDelayCal or Timer2InitDelayNoCal), the RRM still not confirms the IP_BLE has been granted, then:

- no sequence is started (Radio FSM and transmit/receive blocks not informed),
- no timer management is done (no update / reprogramming of the different timers is done),
- no RAM write back occurs, RAM tables are left unchanged,
- the SEMATIMEOUTERROR bit is set in the STATUSREG IP_BLE APB register is set,
- if the IntSemaTimeoutError bit in the GlobalStatMach word 0x05 is set, the SEMATIMEOUTERROR bit is set in the INTERRUPT1REG IP_BLE APB registers and an IP_BLE interrupt is generated to the CPU
  - The enable mask is readable in INTERRUPT1ENABLEREG. SEMATIMEOUTERROR bit.

Radio FSM not in ACTIVE2 state on time

This error happens if the Radio FSM is not in ACTIVE2 state at the end of the 1st INIT step (on ***InitDelay*** timeout expiration).

If at the end of the initialization delay (WakeupInitDelay or Timer12InitDelayCal or Timer2InitDelayNoCal), the Radio FSM still not confirmed it is ready to start a TX or RX sequence (no accurate clock present for instance):

- no sequence is started (Radio FSM and transmit/receive blocks not informed),
- no timer management is done (no update / reprogramming of the different timers is done),
- no RAM write back occurs, RAM tables are left unchanged,
- the ACTIVE2ERROR bit is set in the STATUSREG IP_BLE APB register is set.
- if the IntActive2Error bit in the GlobalStatMach word 0x05 is set, the ACTIVE2ERROR bit is set in the INTERRUPT1REG IP_BLE APB registers and an IP_BLE interrupt is generated to the CPU
  - The enable mask is readable in INTERRUPT1ENABLEREG. ACTIVE2ERROR bit.

Configuration error

A configuration error occurs If the value contained by the Rcvpoint or Txpoint field or IQSamplingPtr of the current StatMach RAM table is not modulo 4 (does not correspond to a 32-bit aligned address).

Note: The StatMach.IQSamplingPtr value is checked only for some values in other RAM table bit fields. Refer to IQSamplesPtr[31:0] or AntennaPatternPtr[31:0] not 32-bit aligned for details.

In this case:

- the sequencer stops the sequence at the end of the 1st initialization phase (so neither the Radio FSM nor the transmit/receive blocks received any request),
- no timer management is done (no update / reprogramming of the different timers),
- no RAM write back occurs,
- the STATUSREG.CONFIGERROR bit is set,
- if the IntConfigError bit in the GlobalStatMach RAM table is set, the INTERRUPT1REG.CONFIGERROR is set and an IP_BLE interrupt is generated,
  - The enable mask is readable in INTERRUPT1ENABLEREG.CONFIGERROR bit.

Address Pointer error

An address pointer error occurs if the TxRxPack.NextPtr[31:24], the TxRxPack.DataPtr[31:24] or the StatMach.IQSamplestPtr[31:24] (if TxRxPack.CTEAndSamplingEnable=1) is not equal to the SoC RAM base address.

In this case:

- No transmission or reception is started,
- the Tx or Rx request towards Radio FSM is canceled,
- no RAM write back is done,
- the STATUSREG.ADDPOINTERROR bit is high at the end of the sequence,
- if the IntAddPointError bit in the GlobalStatMach is high, the INTERRUPT1REG.ADDPOINTERROR bit is set and an IP_BLE interrupt is generated.
  - The enable mask is readable in INTERRUPT1ENABLEREG.ADDPOINTERROR bit.

PLL lock fail (only if GlobalStatMach.AutoTxRxSkipEn = 1)

If the AutoTxRxSkipEn bit in the GlobalStatMach RAM table, the sequencer skips the sequence if the PLL lock fail information is raised.

This PLL lock fail corresponds to the fact the PLL did not lock on time and is provided by the Radio FSM. It is checked by the sequencer at the end of the initialization step, before to enter the TX-RX step.

In this case:

- No transmission or reception is started,
- the Tx or Rx request towards Radio FSM is canceled,
- no RAM write back is done,
- the STATUSREG.TXRXSKIP bit is high at the end of the sequence,
- if the IntTxRxSkip bit in the GlobalStatMach is high, the INTERRUPT1REG.TXRXSKIP bit is set and an IP_BLE interrupt is generated
  - The enable mask is readable in INTERRUPT1ENABLEREG.TXRXSKIP bit.

TxRxSkip APB command

An APB command is available to skip an on-going TX or RX transfer. The software needs to write ‘1’ in TXRXSKIP bit of the CMDREG IP_BLE APB register.

Note: This bit is auto-cleared immediately by the hardware.

The software must be aware the TxRxSkip APB command is considered only during a sequence. Else the skip request is ignored (not recorded and no TXRXSKIP interrupt/status flag is raised on the next sequence).

The behavior differs depending on when the TxRxSkip command occurs inside the sequence.

Table 130. Summary of flags and RAM table pointers behavior versus TX Skip command

“Skip at” phase	Interrupt flags			RAM table pointers updated			RAM Write back
	DONE Bit[25]	TXRXSKIP Bit[21]	TXERROR_1 Bit[9]	TX Prev	TX	TX Next	
1st INIT	NO	YES	NO	NO	NO	NO	NO
2nd INIT	NO	YES	NO	NO	NO	NO	NO
DATA INIT	NO	YES	NO	NO	NO	NO	NO
TX	YES	YES	YES	NO	NO	YES	YES
CONTECT SAVING	YES	NO	NO	YES	YES	YES	YES

Table 131. Summary of flags and RAM table pointers behavior versus RX Skip command

“Skip at” phase	Interrupt flags				RAM table pointers updated		RAM Write back
	RCVOK Bit[31]	RCVCRCERR Bit[30]	DONE Bit[25]	TXRXSKIP Bit[21]	RCV Prev	RCV	
1st INIT	NO	NO	NO	YES	NO	NO	NO
2nd INIT	NO	NO	NO	YES	NO	NO	NO
DATA INIT	NO	NO	NO	YES	NO	NO	NO
RX	NO	YES	YES	YES	NO	YES	YES
CONTECT SAVING	YES	NO	YES	NO	YES	YES	YES

In all scenarios, the IntTxRxSkip bit in the GlobalStatMach must be high to have an interrupt generated when STATUSREG.TXRXSKIP bit is high. In this case, INTERRUPT1REG.TXRXSKIP bit is also high. The enable mask is readable in INTERRUPT1ENABLEREG.TXRXSKIP bit.

AllTableReady bit not set on time

During the 2nd INIT step, the sequencer reads the TxRxPack.AllTableReady bit just after the ConfigEndDuration waiting loop (but checks its value only when exiting DATA INIT step at the end of init_radio_delay).

The role of this bit is to guarantee the information related to the transmission/reception packet (especially data pointers) in the TxRxPack (except the payload bytes when transmission) are valid/up-to-date.

If the recorded TxRxPack.AllTableReady is not high:

- No transmission or reception is started,
- Tx or Rx request towards Radio FSM is canceled,
- no RAM write back is done,
- the STATUSREG.ALLTABLEREADYERROR bit is high at the end of the sequence,
- if IntAllTableReadyError bit in the GlobalStatMach is high, the INTERRUPT1REG.ALLTABLEREADYERROR bit is set and an IP_BLE interrupt is generated
  - The enable mask is readable in INTERRUPT1ENABLEREG.ALLTABLEREADYERROR bit.

TxdataReady bits not set on time

This bit is used only when the on-going transfer is a transmission (not checked on a reception).

It adds flexibility to the software to be able to go on filling the data payload to transmit while the transmission has already started on the antenna.

The recommendation is to set this bit only after at least 16 bytes of Tx data payload are available in the data buffer.

The sequencer reads the TxRxPack.TxDataReady bit at the beginning of the DATA INIT step but checks its value only at the end of the init_radio_delay.

If the recorded TxRxPack.TxDataReady is not high:

- No transmission is started,
- Tx request towards Radio FSM is canceled,
- no RAM write back is done,
- the STATUSREG.TXDATAREADYERROR bit is high at the end of the sequence,
- if IntTxDataReadyError bit in the GlobalStatMach is high, the INTERRUPT1REG.TXDATAREADYERROR bit is set and a IP_BLE interrupt is generated
  - The enable mask is readable in INTERRUPT1ENABLEREG.TXDATAREADYERROR bit.

Receive length error

This aborting issue can occur only on reception.

A Receive length error is detected if the received length value decoded in the received frame header is greater than the StatMach.MaxReceivedLength[7:0] bit field. This feature can be used when free RAM area is limited and does not allow receiving big packets.

If the receive block detect a packet length in the received header that is greater than the StatMach.MaxReceiveLength value:

- the receiver treats only MaxReceiveLength bytes of data and stops its sequence
  - any extra bytes sent by the demodulator is ignored,
- the data payload written back in RAM is limited to MaxReceiveLength + 2 bytes (corresponding to header + length) + potential CTEINFO byte in case of Data PDU channel with CTE present
- the sequencer manages the reception step as usual as the receive block provides a done pulse,
  - this leads to stop the Radio FSM receive mode while data are potentially still arriving on the antenna
- a RAM write back occurs,
- the STATUSREG.RCVLENGTHERROR bit is high at the end of the sequence,
- if the IntRcvLengthError bit in the GlobalStatMach is high, the INTERRUPT1REG.RCVLENGTHERROR bit is set and an IP_BLE interrupt is generated
  - The enable mask is readable in INTERRUPT1ENABLEREG.RCVLENGTHERROR bit.

Note: As the receiver truncates the received frame to a reduced length, a CRC error occurs in parallel and potential other side effect error flags. So the RCVLENGTHERROR flag must be considered before the others regarding a reception.

25.9.3 IP_BLE interrupts

The Bluetooth LE link layer provides 2 separate interrupt lines:

- int1: Bluetooth LE sequence interrupt (linked to sequence that just occurred)
- int2: AES interrupt (manual or LE privacy end of calculation)

The IP_BLE interrupts:

- the IP_BLE interrupts are enabled through the RAM tables (some in the GlobalStatMach, others in TxRxPack),
- a copy of the applied enable mask is available in the INTERRUPT1ENABLEREG IP_BLE APB register,
- the IP_BLE interrupts status can be read when an interrupt trigs at CPU level and is available in the INTERRUPT1REG IP_BLE APB register,
- the IP_BLE interrupts are cleared by writing ‘1’ in the associated bit in the INTERRUPT1REG IP_BLE APB register.
- a dedicated internal timer is started when the interrupt is raised: its value is accessible in the INTERRUPT1LATENCYREG register. The goal of this register is to inform the interrupt handler SW about from how long the interrupt is pending. This latency window is limited to 255 µs. Above this delay, the read value stays at 255.

Note: Only enabled interrupts can be read at '1' in this INTERRUPT1REG register. To have the equivalent without enable mask, the STATUSREG IP_BLE APB register must be read.

The AES interrupts:

- the AES interrupts are enabled through IP_BLE APB registers (MANAESCMDREG for manual AES and AESLEPRIVCMDREG for LE privacy AES),
- the AES interrupts status can be read in the INTERRUPT2REG IP_BLE APB register and represents/means "end of calculation" information,
- the AES interrupts are cleared by writing '1' in the associated bit in the INTERRUPT2REG IP_BLE APB register.

Note: The "end of calculation" information is only available through enabled interruption. The MANAESSTATREG and the AESLEPRIVSTATREG IP_BLE APB registers contain other flags, not equivalent to INTERRUPT2REG registers.

25.9.4 IP_BLE RAM table

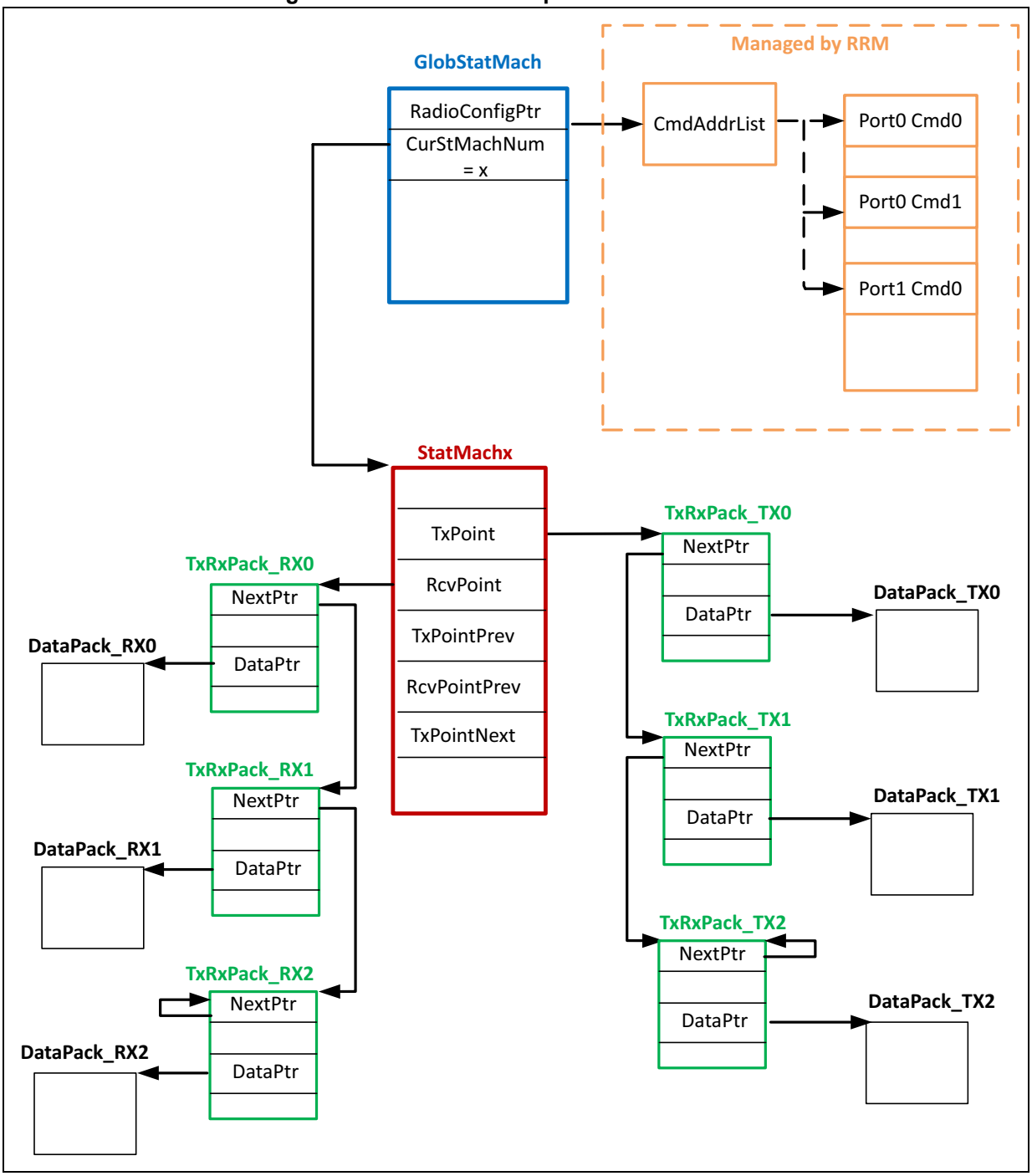
Each time a trigger event is sent to the Bluetooth LE link layer, the sequencer fetches the RAM tables to get the needed information to know what to configure for the radio and which sequence to start (RX or TX)

There are several types of tables:

- The GlobalStatMach: this table is unique.
- The StatMach: one table by active connection (up to 128 supported by the hardware).
- The TxRxPack: one table packet in RX or in TX. There is no predefined number of those tables. They are used as link list from one packet to another during a full connection.
- The DataPack tables corresponding to the data buffers pointed by the DataPtr in the TxRxPack. It contains the PDU section of the Bluetooth packet.

Table 205 gives an overview of RAM tables dependencies. In the provided example, the GlobalStatMach is currently managing the connection number X (StatMachx on-use) and the data buffer points on itself from the third transfer of each type.

Figure 205. RAM tables dependencies overview



GlobalStatMach RAM table

The GlobalStatMach location is frozen by the hardware at address 0x2000_00C0 in the STM32WB09xE product. Refer to *Table 4: SRAM0 reserved locations*.

GlobalStatMach RAM table overview

The GlobalStatMach is unique and mainly contains static information/options.

Table 132. GlobalStatMach RAM table(1)

Word	Byte addr	R/W by IP_BLE	7	6	5	4	3	2	1	0
0x00	0x0	No	RadioConfigPtr[7:0]							
	0x1		RadioConfigPtr[15:8]							
	0x2		RadioConfigPtr[23:16]							
	0x3		RadioConfigPtr[31:24]							
0x01	0x4	R/W	Active	CurStMachNum						
	0x5		WakeupInitDelay							
	0x6		Timer12InitDelayCal							
	0x7		Timer2InitDelayNoCal							
0x02	0x8	R	TransmitCalDelayChk							
	0x9		TransmitNoCalDelayChk							
	0xA		ReceiveCalDelayChk							
	0xB		ReceiveNoCalDelayChk							
0x03	0xC	R	ConfigEndDuration							
	0xD		TxdataReadyCheck							
	0xE		TxdelayStart							
	0xF		TimeCapture(2)	TimeCaptureSel(2)	TxdelayEnd					
0x04	0x10	R	TxReadyTimeout							
	0x11		RcvTimeout[7:0]							
	0x12		RcvTimeout [15:8]							
	0x13		-	-	-	-	RcvTimeout [19:16]			

Table 132. GlobalStatMach RAM table(1) (continued)

Word	Byte addr	R/W by IP_BLE	7	6	5	4	3	2	1	0
0x05	0x14	R	-	-	-	-	-	ChkFlagAutoClearEna		AutoTxRxskipEn
	0x15		-	-	-	-	-	-	-	-
	0x16		IntNoActiveLError	IntTxDataReadyError	IntAllTableReadyError	IntAddPointError	-	-	-	-
	0x17		IntConfigError	IntActive2Err	intTxRxSkip	IntSeqDone	-	IntSemaTimeoutError	IntRcvLengthError	-
0x06	0x18	R	-	DefaultAntennaID[6:0]						
	0x19		-	-	-	-	-	-	-	-
	0x1A		-	-	-	-	-	-	-	-
	0x1B		-	-	-	-	-	-	-	-

1. Unused cells are filled with “-”
2. Content of pink-shaded cells is for debug and qualification purposes.

The following sub-chapter describes the GlobalStatMach bit fields to help the user to program accurately the table.

init_radio_delay is the generic name for the delay that can be TransmitCalDelayChk or TransmitNoCalDelayChk or ReceiveCalDelayChk or ReceiveNoCalDelayChk depending on transfer configuration (see Second initialization step for more details).

25.10 GlobalStatMach registers

Table 133. GLOBALSTATMACH_REG_BLOCK register list

Offset	Register Name	Register Title	Reset
0x0000	WORD0	WORD0 register	0x0000 0000
0x0004	WORD1	WORD1 register	0x0000 0000
0x0008	WORD2	WORD2 register	0x0000 0000
0x000C	WORD3	WORD3 register	0x0000 0000
0x0010	WORD4	WORD4 register	0x0000 0000
0x0014	WORD5	WORD5 register	0x0000 0000
0x0018	WORD6	WORD6 register	0x0000 0000

25.10.1 WORD0 register (WORD0)

Address offset: 0x0000

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RADIOCONFIGPTR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RADIOCONFIGPTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RADIOCONFIGPTR[31:0]**: Radio Configuration address Pointer.

Contains the address of the command_start_list used by the RRM block to execute UDRA command.

Caution: This pointer must be 32-bit aligned.

Note: this value is loaded automatically by the RRM when the MR_BLE_V2 IP exits reset. However, it is also possible to make the RRM reload it through a reload command in UDRA_CTRL register

25.10.2 WORD1 register (WORD1)

Address offset: 0x0004

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMER2INITDELAYNOCAL[7:0]								TIMER12INITDELAYCAL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEUPINITDELAY[7:0]								ACTIVE	CURSTMACHNUM[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **TIMER2INITDELAYNOCAL[7:0]**: Delay between Timer2 trig event on sequencer and RX/TX request sending to the Radio FSM. It corresponds to the sequencer 1st INIT step duration.

This bit field is used for Timer2 trig event only if CalReq bit is low in current TxRxPack RAM table (No PLL Calibration is requested).

The time unit for this delay is 1 µs.

Note: This field is written back with value read at the beginning of the sequence only if the ChkFlagAutoClearEna bit = '1'.

Bits 23:16 **TIMER12INITDELAYCAL[7:0]**: Delay between Timer1 or Timer2 trig event on sequencer and RX/TX request sending to the Radio FSM. It corresponds to the sequencer 1st INIT step duration.

This bit field is used for Timer2 trig event only if CalReq bit is set in current TxRxPack RAM table (PLL Calibration is requested.

The time unit for this delay is 1 µs.

Note: This field is written back with value read at the beginning of the sequence only if the ChkFlagAutoClearEna bit = ‘1’.

Bits 15:8 **WAKEUPINITDELAY[7:0]**: Delay between wakeup timer trig event on sequencer and RX/TX request sending to the Radio FSM. It corresponds to the sequencer 1st INIT step duration.

Note: this bit field is not used if trig event comes from Timer1 or Timer2.

The time unit for this delay/value is a period of slow clock frequency x 16 (if slow clock is 32kHz, this bit field unit is 1 period of 512 kHz).

Note: This field is written back with value read at the beginning of the sequence only if the ChkFlagAutoClearEna bit = '1'.

Bit 7 **ACTIVE**: Must be at '1' when the trig event (Wakeup Timer, Timer1 or Timer2) occurs to starts a Bluetooth LE link layer sequence. Else no RF sequence nor timer management is done by the sequencer.

Only SEQDONE and NOACTIVELERROR flags are raised in STATUSREG and INTERRUPT1REG (if associated interrupts are enabled).

Note: This field is written back to '0' only if the ChkFlagAutoClearEna bit = '1'.

Bits 6:0 **CURSTMACHNUM[6:0]**: current connection machine number.

Defines the state machine number (in the range from 0 to 127) which is running for the current transmission or reception.

It is used to calculate the RAM address from which the State machine table (“StateMach”) is read.

Note: This field is written back with value read at the beginning of the sequence only if the ChkFlagAutoClearEna bit = '1'.

25.10.3 WORD2 register (WORD2)

Address offset: 0x0008

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD2 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RECEIVENOCALDELAYCHK[7:0]								RECEIVECALDELAYCHK[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSMITNOCALDELAYCHK[7:0]								TRANSMITCALDELAYCHK[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **RECEIVENOCALDELAYCHK[7:0]**: Delay between RX request sent to the Radio FSM and the start pulse to the receive block. It corresponds to the sequencer 2nd INIT + DATA INIT steps duration.

Note: this bit field is used if TxMode bit is low in the StatMach (reception) and the CalReq bit is low in current TxRxPack RAM table (no PLL Calibration is requested).

The time unit for this delay is 1 µs.

Bits 23:16 **RECEIVECALDELAYCHK[7:0]**: Delay between RX request sent to the Radio FSM and the start pulse sent to the receive block. It corresponds to the sequencer 2nd INIT + DATA INIT steps duration.

Note: this bit field is used if TxMode bit is low in the StatMach (reception) and the CalReq bit is set in current TxRxPack RAM table (PLL Calibration is requested).

The time unit for this delay is 1 µs.

Bits 15:8 **TRANSMITNOCALDELAYCHK[7:0]**: Delay between TX request sent to the Radio FSM and the start pulse to the transmit block. It corresponds to the sequencer 2nd INIT + DATA INIT steps duration.

Note: this bit field is used if TxMode bit is set in the StatMach (transmission) and the CalReq bit is low in current TxRxPack RAM table (no PLL Calibration is requested).

The time unit for this delay is 1 µs.

Bits 7:0 **TRANSMITCALDELAYCHK[7:0]**: Delay between TX request sent to the Radio FSM and the start pulse sent to the transmit block. It corresponds to the sequencer 2nd INIT + DATA INIT steps duration.

Note: this bit field is used if TxMode bit is set in the StatMach (transmission) and the CalReq bit is set in current TxRxPack RAM table (PLL Calibration is requested).

The time unit for this delay is 1 µs.

25.10.4 WORD3 register (WORD3)

Address offset: 0x000C

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD3 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMECAPTURE	TIMECAPTURESEL	TXDELAYEND[5:0]						TXDELAYSTART[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDATAREADYCHECK[7:0]								CONFIGENDDURATION[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMECAPTURE**:

0: No capture is requested to monitor the Bluetooth LE sequence.

1: a time capture is requested to monitor the Bluetooth LE sequence. Captured event is defined by GlobalStatMach.TIMECAPTURESEL bit.

Note: if both TIMECAPTURE and TIMECAPTURESEL bits are low, the TimerCaptureReg IP_BLE APB register is anyway updated with the InitDelay timeout event (mechanism to bypass the fact those 2 GlobalStatMach bits are checked after 1st INIT step completion).

Note: If TxRxPack.TrigRcv or TxRxPack.TrigDone bit is set, the TimerCaptureReg IP_BLE APB register shows this last event trig value at the end.

Note: this bit is for debug purpose.

Bit 30 **TIMECAPTURESEL**:

0: the captured time (absolute time) corresponds to the end of 1st INIT step in the sequence (InitDelay timeout event).

1: the captured time (absolute time) corresponds to the end of DATA INIT step in the sequence (init_radio_delay timeout event).

Note: this bit is for debug purpose.

Bits 29:24 **TXDELAYEND[5:0]**: Delay added between the last bit transmission to the modulator and the “end of transmission”information for the sequencer.

The time unit for this delay is 125 ns.

This delay allows giving time to the modulator and analog chain to output on the antenna the last bit.

Bits 23:16 **TXDELAYSTART[7:0]**: Delay added between the moment the Radio FSM is in TX mode (PA ramp up done and power present on the antenna) and the first bit transmission to the modulator.

The time unit for this delay is 125 ns.

Bits 15:8 **TXDATAREADYCHECK[7:0]**: Duration for the sequencer to get the TxDataReady and DataPtr information in TxRxPack table.

The goal of this bit field is to provide more time to the Firmware to provide the data pointer address and in case of transmission to provide the data to transmit.

The sequencer waits for relative time to be equal to init_radio_delay –TxdataReadyCheck before to start the final configuration.

The time unit for this delay is 1 µs.

Bits 7:0 **CONFIGENDDURATION[7:0]**: Duration for the sequencer to execute the final configuration.

The goal of this bit field is to provide more time to the Firmware to prepare the RAM tables.

The sequencer waits for relative time to be equal to init_radio_delay –ConfigEndDuration before to start the final configuration.

The time unit for this delay is 1µs.

25.10.5 WORD4 register (WORD4)

Address offset: 0x0010

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD4 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RCVTIMEOUT[19:8]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCVTIMEOUT[7:0]								TXREADYTIMEOUT[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:8 **RCVTIMEOUT[19:0]**: Receive window timeout.

Define the maximum duration to stay in reception without any preamble + access address detection (rest of the frame can be received even outside this time window).
The duration is expressed as (4^RCVTIMEOUT[19;18]) x RCVTIMEOUT[17:0]
The time unit for RCVTIMEOUT[17:0] is 1 µs.

Bits 7:0 **TXREADYTIMEOUT[7:0]**: Transmission ready timeout.

Defines the maximum duration for the transmit block to wait for the Radio FSM to indicate it is in TX state and data can be provided to the modulator.
The time unit for this delay is 1 µs.
Note: if this value is set to 0, no timeout is activated to wait the TX ready information. This configuration is not recommended at all as it may lead to endless sequence, restarted only through a new trigger event is generated

25.10.6 WORD5 register (WORD5)

Address offset: 0x0014

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD5 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTCONFIGERROR	INTACTIVE2ERR	INTTXRXSKIP	INTSEQDONE	Res.	INTSEMATIMEOUTERROR	INTRCVLENGTHERROR	Res.	INTNOACTIVELERROR	INTTXDATAREADYERROR	INTALLTABLEREADYERROR	INTADDPOINTERROR	Res.	Res.	Res.	Res.
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHKFLAGAUTOCLEARENA	Res.	AUTOTXRXSKIPEN
													rw		rw

Bit 31 **INTCONFIGERROR**: Configuration error interrupt enable.
0: the interrupt associated to INTERRUPT1REG.ConfigError is disabled
1: the interrupt associated to INTERRUPT1REG. ConfigError is enabled.

Bit 30 **INTACTIVE2ERR**: not “in ACTIVE2”information from Radio FSM received on time interrupt enable.
0: the interrupt associated to INTERRUPT1REG.Active2Error is disabled.
1: the interrupt associated to INTERRUPT1REG.Active2Error is enabled.

Bit 29 **INTTXRXSKIP**: Transmission or reception skip interrupt enable.
0: the interrupt associated to INTERRUPT1REG.intTxRxSkip is disabled.
1: the interrupt associated to INTERRUPT1REG.intTxRxSkip is enabled.

Bit 28 **INTSEQDONE**: sequencer end of task interrupt enable.
0: the interrupt associated to INTERRUPT1REG.SeqDone is disabled.
1: the interrupt associated to INTERRUPT1REG.SeqDone is enabled.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **INTSEMATIMEOUTERROR**: Semaphore timeout error interrupt enable.
0: the interrupt associated to INTERRUPT1REG.SemaTimeoutError is disabled.
1: the interrupt associated to INTERRUPT1REG.SemaTimeoutError is enabled.

Bit 25 **INTRCVLENGTHERROR**: Too long received payload length interrupt enable.
0: the interrupt associated to INTERRUPT1REG.ReceiveLengthError is disabled.
1: the interrupt associated to INTERRUPT1REG.ReceiveLengthError is enabled.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **INTNOACTIVELERROR**: Active bit low value reading interrupt enable.
0: the interrupt associated to INTERRUPT1REG.NoActiveLError is disabled.
1: the interrupt associated to INTERRUPT1REG.NoActiveLError is enabled.

Bit 22 **INTTXDATAREADYERROR**: Transmission data payload ready error interrupt enable.
0: the interrupt associated to INTERRUPT1REG.TxDataReady is disabled
1: the interrupt associated to INTERRUPT1REG.TxDataReady is enabled.

Bit 21 **INTALLTABLEREADYERROR**: All table ready error interrupt enable.
0: the interrupt associated to INTERRUPT1REG.AllTableReadyError is disabled.
1: the interrupt associated to INTERRUPT1REG.AllTableReadyError is enabled.

Bit 20 **INTADDPOINTERROR**: Address pointer error interrupt enable.
0: the interrupt associated to INTERRUPT1REG.AddPointError is disabled.
1: the interrupt associated to INTERRUPT1REG.AddPointError is enabled.

Bits 19:3 Reserved, must be kept at reset value.

Bit 2 **CHKFLAGAUTOCLEARENA**: Active bit Auto Clear Enable.
The Active auto clear feature leads the sequencer to clear the GlobalStatMach.Active bit during the RAM write back step at the end of a transfer/sequence.
Main goal of this feature is to avoid a new transfer to start on the antenna while the software did not yet prepare the next transfer in RAM tables.
0: The active auto clear bit feature is disabled.
1: The active auto clear bit feature is enabled.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **AUTOTXRXSKIPEN**: Automatic transfer (TX or RX) skip enable.
If set, the Bluetooth LE link layer stops automatically an on-going transfer is PLL lock fail event is detected on PLL start.

25.10.7 WORD6 register (WORD6)

Address offset: 0x0018

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD6 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEFAULTANTENNAID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **DEFAULTANTENNAID[6:0]**: Default Antenna ID corresponding to the number of the antenna used to receive/transmit:

- the full packet when no CTE
- the packet body (Preamble, Access Address, PDU and CRC) when CTE

25.10.8 StatMach RAM table

StatMach RAM table overview

The StatMach table is linked to an active connection. There is as many StatMach table as concurrent connections in a limit of 128 (maximum supported by the hardware).

The StatMach RAM tables location is frozen by the hardware as chained just after the GlobalStatMach. The formula for a StatMach base address is:

$$StatMachBaseAddress[StatMachIdx] = GlobalStatMachBaseAddress + 28 + (StatMachIdx * 92)$$

Table 134. StatMach RAM table(1)

Word	Byte addr	R/W by IP_BLE	7	6	5	4	3	2	1	0
0x00	0x00	R/W	TxMode	RadioComListEnaUchan						
	0x01		NESN	SN	Remap_chan					
	0x02		RcvEnc	TxEnc	EncryptOn	Buffer_Full	-	-	PhysChanPDUType[1:0]	
	0x03		-	RxPhy[2:0]			CTEDisable	TxPhy[2:0]		
0x01	0x04	R/W	Txpoint[7:0]							
	0x05		Txpoint[15:8]							
	0x06		Txpoint[23:16]							
	0x07		Txpoint[31:24]							
0x02	0x08	R/W	Rcvpoint[7:0]							
	0x09		Rcvpoint[15:8]							
	0x0A		Rcvpoint[23:16]							
	0x0B		Rcvpoint[31:24]							
0x03	0x0C	R/W	TxpointPrev[7:0]							
	0x0D		TxpointPrev[15:8]							
	0x0E		TxpointPrev[23:16]							
	0x0F		TxpointPrev[31:24]							

Table 134. StatMach RAM table(1) (continued)

Word	Byte addr	R/W by IP_BLE	7	6	5	4	3	2	1	0
0x04	0x10	R/W	RcvpointPrev[7:0]							
	0x11		RcvpointPrev[15:8]							
	0x12		RcvpointPrev[23:16]							
	0x13		RcvpointPrev[31:24]							
0x05	0x14	R/W	Txpointnext[7:0]							
	0x15		Txpointnext[15:8]							
	0x16		Txpointnext[23:16]							
	0x17		Txpointnext[31:24]							
0x06	0x18	R/W	PCntTx[7:0]							
	0x19		PCntTx [15:8]							
	0x1A		PCntTx [23:16]							
	0x1B		PCntTx [31:24]							
0x07	0x1C	R/W	PCntTx [39:32]							
	0x1D		PCntRcv[7:0]							
	0x1E		PCntRcv[15:8]							
	0x1F		PCntRcv[23:16]							
0x08	0x20	R	PCntRcv[31:24]							
	0x21		PCntRcv[39:32]							
	0x22		RxMicDbg(2)	MsbFirst (3)	DisableCrc(3)	EnaPreambleRep(3)	PreambleRep[3:0](3)			
	0x23		RxDebugCrc(2)	IntRxOverflowError	IntEncError	intTxError[4:0]				
0x09	0x24	R	accaddr[7:0]							
	0x25		accaddr[15:8]							
	0x26		accaddr[23:16]							
	0x27		accaddr[31:24]							
0x0A	0x28	R	crcinit[7:0]							
	0x29		crcinit[15:8]							
	0x2A		crcinit[23:16]							
	0x2B		MaxReceivedLength							

Table 134. StatMach RAM table(1) (continued)

Word	Byte addr	R/W by IP_BLE	7	6	5	4	3	2	1	0
0x0B	0x2C	R	TxHp	-	-	PaPower				
	0x2D		-	-	hopincr					
	0x2E		UsedChannelFlags[7:0]							
	0x2F		UsedChannelFlags[15:8]							
0x0C	0x30	R	UsedChannelFlags[23:16]							
	0x31		UsedChannelFlags[31:24]							
	0x32		-	-	-	UsedChannelFlags[36:32]				
	0x33		-							
0x0D	0x34	R	eventCounter[7:0]							
	0x35		eventCounter[15:8]							
	0x36		-							
	0x37		-							
0x0E	0x38	R	EncryptIV[7:0]							
	0x39		EncryptIV[15:8]							
	0x3A		EncryptIV[23:16]							
	0x3B		EncryptIV[31:24]							
0x0F	0x3C	R	EncryptIV[39:32]							
	0x3D		EncryptIV[47:40]							
	0x3E		EncryptIV[55:48]							
	0x3F		EncryptIV[63:56]							
0x10	0x40	R	EncryptK[7:0]							
	0x41		EncryptK[15:8]							
	0x42		EncryptK[23:16]							
	0x43		EncryptK[31:24]							
0x11	0x44	R	EncryptK[39:32]							
	0x45		EncryptK[47:40]							
	0x46		EncryptK[55:48]							
	0x47		EncryptK[63:56]							
0x12	0x48	R	EncryptK[71:64]							
	0x49		EncryptK[79:72]							
	0x4A		EncryptK[87:80]							
	0x4B		EncryptK[95:88]							

Table 134. StatMach RAM table(1) (continued)

Word	Byte addr	R/W by IP_BLE	7	6	5	4	3	2	1	0
0x13	0x4C	R	EncryptK[103:96]							
	0x4D		EncryptK[111:104]							
	0x4E		EncryptK[119:112]							
	0x4F		EncryptK[127:120]							
0x14	0x50	R	-	CTETime[4:0]					CTESlot Width	AoD_nAoA
	0x51		-	MaximumIQSamplesNumber[6:0]						
	0x52		AntennaPatternLength[7:0]							
	0x53		-							
0x15	0x54	R	IQSamplesPtr[7:0]							
	0x55		IQSamplesPtr[15:8]							
	0x56		IQSamplesPtr[23:16]							
	0x57		IQSamplesPtr[31:24]							
0x16	0x58	R	AntennaPatternPtr[7:0]							
	0x59		AntennaPatternPtr[15:8]							
	0x5A		AntennaPatternPtr[23:16]							
	0x5B		AntennaPatternPtr[31:24]							

1. Unused cells are filled with “-”.
2. Content of pink-shaded cells is for debug and qualification purposes.
3. Content of light-blue shaded cells is related to features outside Bluetooth protocol (proprietary protocol).

25.11 StatMach registers

This section describes the StatMach bit fields to help the user to program accurately the table.

Table 135. STATMACH_REG_BLOCK register list

Offset	Register Name	Register Title	Reset
0x0000	WORD0	WORD0 register	0x0000 0000
0x0004	WORD1	WORD1 register	0x0000 0003
0x0008	WORD2	WORD2 register	0x0000 0000
0x000C	WORD3	WORD3 register	0x0000 0000
0x0010	WORD4	WORD4 register	0x0000 0000
0x0014	WORD5	WORD5 register	0x0000 0000
0x0018	WORD6	WORD6 register	0x0000 0000
0x001C	WORD7	WORD7 register	0x0000 0000
0x0020	WORD8	WORD8 register	0x0000 0000
0x0024	WORD9	WORD9 register	0x0000 0000
0x0028	WORDA	WORDA register	0x0000 0000
0x002C	WORDB	WORDB register	0x0000 0000
0x0030	WORDC	WORDC register	0x0000 0000
0x0034	WORDD	WORDD register	0x0000 0000
0x0038	WORDE	WORDE register	0x0000 0000
0x003C	WORDF	WORDF register	0x0000 0000
0x0040	WORD10	WORD10 register	0x0000 0000
0x0044	WORD11	WORD11 register	0x0000 0000
0x0048	WORD12	WORD12 register	0x0000 0000
0x004C	WORD13	WORD13 register	0x0000 0000
0x0050	WORD14	WORD14 register	0x0000 0000
0x0054	WORD15	WORD15 register	0x0000 0000
0x0058	WORD16	WORD16 register	0x0000 0000

25.11.1 WORD0 register (WORD0)

Address offset: 0x0000

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXPHY[2:0]			CTEDISABLE	TXPHY[2:0]			RCVENC	TXENC	ENCRYPTON	BUFFER_FULL	Res.	Res.	PHYSCHANPDUTYPE[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NESN	SN	REMAP_CHAN[5:0]						TXMODE	RADIOCOMLISTENA	UCHAN[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **RXPHY[2:0]**: Reception Phy selection.

bit0: bit rate (0=1Mbps / 1=2MBps) / bit1: does not care / bit2: coded/legacy.

- 000: Selected Receiver PHY is legacy 1Mbps
- 001: Selected Receiver PHY is legacy2Mbps
- 1x0: Selected Receiver PHY is coded 1Mbps
- othersReserved for future use. If programmed by mistake, selects “Receiver PHY is legacy 1Mbps”option.

Note: S2/S8 coded choice comes from an auto-detection done by the demodulator.

Bit 27 **CTEDISABLE**: Disable the CTE feature

0:

- in transmission, the CTE is appended to the packet if TxRxPack.CTEAndSamplingEnable bit is set,
- in reception, the CTE detection is active

1: CTE is never appended on a transmitted packet and CTE detection mechanism is not active in reception whatever the rest of the RAM table bit fields linked to CTE.

Bits 26:24 **TXPHY[2:0]**: Transmission Phy selection.

- 000: Selected Transmitter PHY is legacy 1Mbps
- 001: Selected Transmitter PHY is legacy 2Mbps
- 100: Selected Transmitter PHY is coded 1Mbps with S=8
- 110: Selected Transmitter PHY is coded 1Mbps with S=2
- othersReserved for future use. If programmed by mistake, selects “Transmitter PHY is legacy 1Mbps”option.

Bit 23 **RCVENC**: Last Receive packet was encrypted.

Note: This bit is fully managed by the hardware.

It is set to 1 after the reception of a packet with length not zero (whatever the CRC check result) if StatMach.Encryption=1.

When RcvEnc=1, the PCntRcv (receive packet counter required for the sub-keys calculation) is incremented depending on the SN/NESN check result.

Bit 22 **TXENC**: Previous transmission packet was encrypted.

Note: This bit is fully managed by the hardware.

It is set to 1 after the transmission of an encrypted packet (so with length not zero).

When TxEnc=0, PCntTx (transmission packet counter required for the sub-keys calculation) is unchanged.

When TxEnc=,1 PCntTx may be incremented depending on the SN/NESN check result.

Bit 21 **ENCRYPTON**: "On the fly" encryption/decryption engine enable.

0: The "On the fly”encryption/decryption engine is disabled.

1: The "On the fly”encryption/decryption engine is enabled.

The parameters StateMach.EncryptIV and StateMach.EncryptK are read from RAM during the initialization phase.

Note: The "On the fly" encryption/decryption engine does not run for packet with null length.

Note: It is mandatory to have TxRxPack.SN_EN=1 when StateMach.Encryption=1 as PCntTx is incremented by the SN/NESN automatic management mechanism.

Bit 20 **BUFFER_FULL**: (also called BUFFOVERFLOW). No more receive buffer available.

Set this bit to indicate no more buffer is available to receive any packet.

In this case:

- No data is written back in the RAM at the end of the sequence
- The SN/NESN automatic mechanism adapts its behavior by keeping the NESN unchanged and not incrementing the encryption receive packet counter.

Note: The SN bit management is not impacted to keep the transmission progressing as long as the peer acknowledge the reception of previous transmitted packet.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **PHYSCHANPDUTYPE[1:0]**: Define which is the PDU Type exchanged.

This information is used by the AES block to adapt the header mask on Encryption Algorithm depending of the PDU Type.

00: ACL (Asynchronous Connection-oriented)

01: CIS (Connected Isochronous Stream)

10: BIS (Broadcast Isochronous Stream)

11: Reserved

Bit 15 **NESN**: Bluetooth LE next expected sequence number bit.

- If TxRxPack.SN_EN=0 or TxRxPack.Advertise=1, this bit is kept unchanged at the end of a transfer.
- If TxRxPack.SN_EN=1 and TxRxPack.Advertise=0, this bit is managed automatically by the hardware SN/NESN mechanism (as described in the Bluetooth core specification *[2]*). Then, this bit is modified by the hardware only at the end of a reception (not on transmission).

Note: in any case, this bit is written back by the sequencer at the end of a transfer (modified or not)

Bit 14 **SN**: Bluetooth LE sequence number bit

- If TxRxPack.SN_EN=0 or TxRxPack.Advertise=1, this bit is kept unchanged at the end of a transfer.
- If TxRxPack.SN_EN=1 and TxRxPack.Advertise=0, this bit is managed automatically by the hardware SN/NESN mechanism (as described in the Bluetooth core specification *[2]*v). Then, this bit is modified by the hardware only at the end of a reception (not on transmission).

Note: in any case, this bit is written back by the sequencer at the end of a transfer (modified or not)

Bits 13:8 **REMAP_CHAN[5:0]**: Bluetooth LE Remapped channel index.

This is the remapped channel as described in algorithm1 and algorithm2 in Bluetooth core specification *[2]*.

This bit field is used by the hardware to generate the physical channel frequency.

Note: This field is written back at the end of the transfer by the sequencer:

- if TxRxPack.IncChan=0, written back value is the same value,
- if TxRxPack.IncChan=1, written back value is the value modified by one of the two algorithms defined by the Bluetooth core specification and mapped to the used channels list.

Note: the standard requests this bit field to be set to 0 for the first connection event.

Bit 7 **TXMODE**: Transfer type selection of the current sequence

0: requested transfer is a reception. The start address of the TxRxPack packet in which the received data has to be stored is pointed by Rcvpoint.

1: requested transfer is a transmission. The start address of the TxRxPack packet to be transmitted is pointed by TxPoint.

Note: this bit is overloaded by the sequencer with StatMach.NextTxMode bit value during each RAM write back phase.

Bit 6 **RADIOCOMLISTENA**: Radio command list enable.

0: The sequencer does not start an UDRA command to the RRM on a trig event

1: The sequencer starts an UDRA command to the RRM on a trig event.

The command number is related to the timer which triggered (0 for Wakeup timer, 1 for Timer1, 2 for Timer2).

Bits 5:0 **UCHAN[5:0]**: Bluetooth LE Unmapped channel index.

UChan is used by the channel incrementer and the remapper to generate a new Uchan and RemapChan values. through the two algorithms defined by the Bluetooth core specification *[2]*.

Note: This field is written back at the end of the transfer by the sequencer:

- if TxRxPack.IncChan=0, written back value is the same value,
- if TxRxPack.IncChan=1, written back value is the value modified by one of the two algorithms defined by the Bluetooth core specification.

Note: the standard requests this bit field to be set to 0 for the first connection event.

25.11.2 WORD1 register (WORD1)

Address offset: 0x0004

Reset value: 0x0000 0003 (0xFFFF FFFF)

WORD1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPOINT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPOINT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TXPOINT[31:0]**: Pointer to transmit packet.

TxPoint defines the start address of the TxRxPack link list (containing the parameters of the current transmission to be proceeded).

This variable needs to be initialized by the firmware with the start address of first the first TxRxPack of the transmission linked list each time a StateMach is created in memory (new connection). Then, TxPoint is managed by the hardware, considering the firmware has to guaranty the transmission link list is never empty (or pointing to itself).

Note: This pointer address must be 32-bit aligned and is an absolute address (not an offset).

25.11.3 WORD2 register (WORD2)

Address offset: 0x0008

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD2 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCVPOINT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCVPOINT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RCVPOINT[31:0]**: Pointer to receive packet.

Rcvpoint defines the start address of the TxRxPack link list (containing the parameters of the current reception to be proceeded)

This variable needs to be initialized by the firmware with the start address of the first TxRxPack of the reception linked list each time a StateMach is created in memory (new connection). Then, RcvPoint is managed by the hardware, considering the firmware has to guaranty the reception link list is never empty (or pointing to itself).

Note: This pointer address must be 32-bit aligned and is an absolute address (not an offset).

25.11.4 WORD3 register (WORD3)

Address offset: 0x000C

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD3 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPOINTPREV[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPOINTPREV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TXPOINTPREV[31:0]**: Pointer to previous transmit packet.

This variable is fully managed by the hardware. It is recommended to initialize to 0 by the firmware when the StateMach is created in memory (new connection).

TxPointPrev indicates which buffer can be reallocated (as it is now free).

25.11.5 WORD4 register (WORD4)

Address offset: 0x0010

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD4 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCVPOINTPREV[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCVPOINTPREV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RCVPOINTPREV[31:0]**: Pointer to previous receive packet.

This variable is fully managed by the hardware. It is recommended to initialize to 0 by the firmware when the StateMach is created in memory (new connection).

RcvPointPrev indicates which buffer can be reallocated (as it is now free).

25.11.6 WORD5 register (WORD5)

Address offset: 0x0014

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD5 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPOINTNEXT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPOINTNEXT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TXPOINTNEXT[31:0]**: Next transmit pointer.

This variable is fully managed by the hardware. It is recommended to initialize to 0 by the firmware when the StateMach is created in memory (new connection).

TxPointNext indicates the address of the TxRxPack transmit packet to be used once the transmission managed by the TxPoint is done (TxRxPack.NextPtr[31:0]).

The TxPointNext bit field is always updated at the end of a transmission.

Note: At the end of a valid reception with TxRxPack.SN_EN=1 and TxRxPack.Advertise=0, the StatMach.TxPoint is equal to the StatMach.TxPointNext.

25.11.7 WORD6 register (WORD6)

Address offset: 0x0018

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD6 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCNTTX_31_0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCNTTX_31_0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PCNTTX_31_0[31:0]**: CCM encryption transmission packet counter [31:0].

PCntTx is used during the on-the-fly encryption of the transmission data by the AES encryption engine.

For each new connection, Bluetooth protocol requires PCntTx to be initialized by the firmware to the value:

- 40'h8000000000: for Data Channel PDUs sent by the central
- 40'h0000000000: for Data Channel PDUs sent by the peripheral.

Note: It is mandatory to have TxRxPack.SN_EN=1 when StateMach.Encryption=1 as PCntTx is incremented by the SN/NESN automatic management mechanism.

25.11.8 WORD7 register (WORD7)

Address offset: 0x001C

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD7 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCNTRCV_23_0[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCNTRCV_23_0[7:0]								PCNTTX_39_32[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 **PCNTRCV_23_0[23:0]**: CCM encryption Receive Packet counter [23:0].

PCntRcv is used during the on-the-fly encryption of the received data by the AES encryption engine.

For each new connection, Bluetooth protocol requires PCntRcv to be initialized by the firmware to the value:

- 40'h8000000000: for Data Channel PDUs received by the perpipheral.
- 40'h0000000000: for Data Channel PDUs receive by the central.

Note: It is mandatory to have TxRxPack.SN_EN=1 as PCntRcv is incremented by the SN/NESN automatic management mechanism.

Bits 7:0 **PCNTTX_39_32[7:0]**: CCM encryption transmission packet counter [39:32].

PCntTx is used during the on-the-fly encryption of the transmission data by the AES encryption engine.

For each new connection, Bluetooth protocol requires PCntTx to be initialized by the firmware to the value:

- 40'h8000000000: for Data Channel PDUs sent by the central
- 40'h0000000000: for Data Channel PDUs sent by the peripheral.

Note: It is mandatory to have TxRxPack.SN_EN=1 when StatMach.Encryption=1 as PCntTx is incremented by the SN/NESN automatic management mechanism.

25.11.9 WORD8 register (WORD8)

Address offset: 0x0020

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD8 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXDEBUGCRC	INTRXOVERFLOWERROR	INTENCERROR	INTTXERROR[4:0]					RXMICDBG	MSBFIRST	DISABLECRC	ENAPREAMBLEREP	PREAMBLEREP[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCNTRCV_39_24[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXDEBUGCRC**: Debug mode of the CRC in reception

0: The received CRC is not saved with payload in RAM (this is the normal mode)

1: The received CRC is saved with payload in RAM (this is a debug mode).

Warning: the SW has to revert the endianness on the CRC data available in the DataBuffer as the HW stores the value with the same endianness as the PDU.

When set:

- the packet is accepted whatever the CRC, so if CRC error, then the RCVOK flag is set anyway and no CRC error flag is raised,
- The DataPack RAM buffer size must consider the 3 additional CRC bytes.

Bit 30 **INTRXOVERFLOWERROR**: Receive data path overflow error interrupt enable

0: the interrupt INTERRUPT1REG.IntRxOverflowError is disabled

1: the interrupt INTERRUPT1REG.IntRxOverflowError is enabled.

Bit 29 **INTENCERROR**: Receive encryption error interrupt enable.

Note: the CRC check result is not considered by the interrupt enabled by IntEncErr.

0: the receive encryption error interrupt is disabled.

1: the receive encryption error interrupt is enabled (and associated interrupt flag is visible in INTERRUPT1REG.ENCERROR.

The interrupt is active if the MIC of the received message does not match the computed one (while the preamble and the access address are received ok, StateMach.Encryption=1 and the received length is not null).

Note: The CRC check result is not considered for this interrupt.

Bits 28:24 **INTTXERROR[4:0]**: Transmission error interrupt enable.

- If IntTxError[n]=1: an interrupt is generated and associated flag is set in INTERRUPT1REG.TXERROR[n] if a TxError[n] event occurs during the transmission.
- If IntTxError[n]=0: no interrupt nor associated flag in INTERRUPT1REG.TXERROR [n] is available if a TxError[n] event occurs during the transmission.

Note: STATUSREG. TXERROR[n] bit is not impacted and always provide the TxError[n] unmasked information.

Bit 23 **RXMICDBG**: Receive MIC debug.

0: The decrypted MIC (locally computed) is stored in the payload buffer in RAM (at the end of the payload).

1: The received MIC is stored in the payload buffer in RAM (at the end of the payload).

When RXMICDBG bit is set, the RCVOK flag is raised at the end of a reception whatever the MIC error status (so even when a MIC error is detected).

This feature is for debug.

Bit 22 **MSBFIRST**: Most significant bit is transmitted first.

0: The Least Significant Bit of the least significant byte is transmitted first in the frame (as described in Bluetooth LE core specification *[2]*).

1: The Most Significant Bit of the Most significant byte is transmitted first in the frame.

This feature is not Bluetooth standard compatible.

Bit 21 **DISABLECRC**: CRC Disable.

If set, this bit:

- in reception: disable the check of the CRC
- in transmission: no CRC field is generated nor inserted in the sent packet.

This feature is not Bluetooth standard compatible.

Warning: when DisableCRC is set, a CRC error flag is systematically set at the end of a reception. Note that the SW is not supposed to track this flag in this configuration.

Bit 20 **ENAPREAMBLEREP**: Enable transmission preamble repetition

0: the preamble feature is disabled and the preamble length is as described in the core specification *[2]*.

1: The preamble feature is enabled and the preamble length is defined by StateMach.PreambleRep (for coded and uncoded phy).

This feature is not Bluetooth standard compatible.

Note: even if the HW allows this feature with Coded PHY configuration, the combination of those 2 settings must be avoided as it creates issues on long preamble sequence.

Bits 19:16 **PREAMBLEREP[3:0]**: transmission Preamble Repetition number.

Defines the number of repetitions of the transmitted preamble length for coded or uncoded phy. Keep it at 0 to have the Bluetooth LE standard preamble format (1 byte).

Note: if StateMach.EnaPreambleRep=0, this bit field is not considered.

This feature is not Bluetooth standard compatible.

Bits 15:0 **PCNTRCV_39_24[15:0]**: CCM encryption Receive Packet counter [39:24].

PCntRcv is used during the on-the-fly encryption of the received data by the AES encryption engine.

For each new connection, Bluetooth protocol requires PCntRcv to be initialized by the firmware to the value:

40'h8000000000: for Data Channel PDUs received by the peripheral.

40'h0000000000: for Data Channel PDUs receive by the central.

Note: It is mandatory to have TxRxPack.SN_EN=1 as PCntRcv is incremented by the SN/NESN automatic management mechanism.

25.11.10 WORD9 register (WORD9)

Address offset: 0x0024

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD9 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ACCADDR[31:0]**: packet access address.

This value is used in transmission and in reception.

- In transmission, it is inserted in the packet after the preamble.
- In reception, it is used by the demodulator to detect and accept a received packet.

Note: The nature of a packet (primary advertising, secondary advertising or data) is only defined by TxRxPack.Advertise so StateMach.Accadr=0x8E89BED6 does not mean that the packet is an advertising packet.

25.11.11 WORDA register (WORDA)

Address offset: 0x0028

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORDA register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAXRECEIVEDLENGTH[7:0]								CRCINIT[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCINIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **MAXRECEIVEDLENGTH[7:0]**: Maximum receive length.

Defines the maximum receive length the Bluetooth LE link layer can accept.

If the length of the received packet is greater than this value, the hardware limits the payload RAM write back data to the defined maximum length and stops the reception treatment on this defined maximum length (implying CRC error, etc.)

The ReceiveLengthError event is raised (visible in STATUSREG and if associated interrupt is enabled in INTERRUPT1REG register).

The received packet is processed normally when the received length located in the received packet header is smaller or equal to StateMach.MaxReceivedLength.

Bits 23:0 **CRCINIT[23:0]**: CRC initialization value.

This value is used to initialize the CRC for Data packet or for AUX_SYNC_IND PDU and its subordinate set.

This bit field is ignored if TxRxPack.CRCINITSEL=0.

25.11.12 WORDB register (WORDB)

Address offset: 0x002C

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORDB register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
USEDCHANNELFLAGS_15_0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HOPINCR[5:0]						TXHP	Res.	Res.	PAPOWER[4:0]				
		rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw

Bits 31:16 **USEDCHANNELFLAGS_15_0[15:0]**: Remapping flags[15:0] for all 37 Bluetooth LE channels.
The remapping flags are used by the Bluetooth smart algorithm 1 and 2.
- If bit(n])=1, the channel n may be used for reception or transmission.
- If bit(n)=0, the channel n cannot be used for reception or transmission.

Note: this parameter is described in Channel Classification/ channel map in the Bluetooth core specification *[2]*.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **HOPINCR[5:0]**: Hop increment.
Defines the hop increment as described in the algorithm 1 of the Bluetooth core specification *[2]*.

Bit 7 **TXHP**: Defines if the transmition is done at maximum output power level.
0: The transmition is not done at maximum output power level
1: The transmition is done at maximum output power level
When this bit is set, the LDO_TRANSFO is bypassed during the transmission and SMPS ouput voltage directly supplies the power amplifier.

Bits 6:5 Reserved, must be kept at reset value.

Bits 4:0 **PAPOWER[4:0]**: Power Amplifier Power.
Defines the transmission output power level expressed in dBm

25.11.13 WORDC register (WORDC)

Address offset: 0x0030

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORDC register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USEDCHANNELFLAGS_36_16[21:16]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USEDCHANNELFLAGS_36_16[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **USEDCHANNELFLAGS_36_16[21:0]**: Remapping flags[36:16] for all 37 Bluetooth LE channels.

The remapping flags are used by the Bluetooth smart algorithm 1 and 2.

If bit(n])=1, the channel n may be used for reception or transmission.

If bit(n)=0, the channel n cannot be used for reception or transmission.

Note: this parameter is described in Channel Classification/ channel map in the Bluetooth core specification [2].

25.11.14 WORDD register (WORDD)

Address offset: 0x0034

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORDD register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENTCOUNTER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EVENTCOUNTER[15:0]**: Event counter value.

Contains a copy of the event counter value, used by the channel incrementer to compute the algorithm

25.11.15 WORDE register (WORDE)

Address offset: 0x0038

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORDE register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENCRYPTIV_31_0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENCRYPTIV_31_0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ENCRYPTIV_31_0[31:0]**: Initial vector for encryption [31:0].
This value is used by the AES engine during on-the-fly AES CCM encryption.
See Bluetooth LE CCM encryption description in Bluetooth LE core specification *[2]*.

25.11.16 WORDF register (WORDF)

Address offset: 0x003C

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORDF register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENCRYPTIV_63_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENCRYPTIV_63_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ENCRYPTIV_63_32[31:0]**: Initial vector for encryption [63:32].
This value is used by the AES engine during on-the-fly AES CCM encryption.
See Bluetooth LE CCM encryption description in Bluetooth LE core specification *[2]*.

25.11.17 WORD10 register (WORD10)

Address offset: 0x0040

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD10 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENCRYPTK_31_0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENCRYPTK_31_0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ENCRYPTK_31_0[31:0]**: Encryption key [31:0].
This value is used by the AES engine during on-the-fly AES CCM encryption.
See Bluetooth LE CCM encryption description in Bluetooth LE core specification *[2]*.

25.11.18 WORD11 register (WORD11)

Address offset: 0x0044

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD11 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENCRYPTK_63_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENCRYPTK_63_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ENCRYPTK_63_32[31:0]**: Encryption key [63:32].
This value is used by the AES engine during on-the-fly AES CCM encryption.
See Bluetooth LE CCM encryption description in Bluetooth LE core specification *[2]*.

25.11.19 WORD12 register (WORD12)

Address offset: 0x0048

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD12 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENCRYPTK_95_64[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENCRYPTK_95_64[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ENCRYPTK_95_64[31:0]**: Encryption key [95:64].
This value is used by the AES engine during on-the-fly AES CCM encryption.
See Bluetooth LE CCM encryption description in Bluetooth LE core specification *[2]*.

25.11.20 WORD13 register (WORD13)

Address offset: 0x004C

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD13 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENCRYPTK_127_96[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENCRYPTK_127_96[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ENCRYPTK_127_96[31:0]**: Encryption key [127:96].
This value is used by the AES engine during on-the-fly AES CCM encryption.
See Bluetooth LE CCM encryption description in Bluetooth LE core specification *[2]*.

25.11.21 WORD14 register (WORD14)

Address offset: 0x0050

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD14 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANTENNAPATTERNLENGTH[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MAXIMUMIQSAMPLESNUMBER[6:0]							Res.	CTETIME[4:0]					CTESLOTWIDTH	AOD_NAOA
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **ANTENNAPATTERNLENGTH[7:0]**: Length of the antenna switching pattern located at address provided by StatMach.AntennaPatternPtr[31:0].

This bit field is used only when TxRxPack.CTEAndSamplingEnable=1 and StatMach.CTEDisable=0.

Note: if the CTE time is longer than the pattern length, the pattern is repeated by the HW as indicated in the Bluetooth LE standard.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **MAXIMUMIQSAMPLESNUMBER[6:0]**: indicates the maximum number of IQ samples that is written during a CTE reception.

If the CTETime leads to more samples, the MR_BLE_V2 stops storing the IQ samples in RAM when this number is reached.

This bit field is used only when StatMach.TxMode=0, TxRxPack.CTEAndSamplingEnable=1 and StatMach.CTEDisable=0.

Note: despite the Bluetooth LE standard specifies the maximum possible number of IQ samples is 82, the MR_BLE offers the possibility to customize the maximum number of IQ sampling to store from 0 to 127.

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **CTETIME[4:0]**: provides to the IP_BLE the duration of the Constant Tone Extension to be appended in transmission mode.

The value is given in 8 µs unit (as the CTETime bit field of the Bluetooth LE standard).

This bit field is used only when StatMach.TxMode=1, TxRxPack.CTEAndSamplingEnable=1 and StatMach.CTEDisable=0.

Bit 1 **CTESLOTWIDTH**: indicates the CTE slot width value.

0: CTE time slot is 1 µs: antenna switching to be done every 2 µs

1: CTE time slot is 2 µs: antenna switching to be done every 4 µs

This bit field is used by the IP_BLE:

- In transmission for AoD feature (StatMach.TxMode=1 and StatMach.AoD_nAoA=1) to control the antenna switching timing.
- In reception for AoA feature (StatMach.TxMode=0 and StatMach.AoD_nAoA=0) to control the antenna switching and IQ sampling timings.

Note:

- in AoD reception, the CTESlotWidth information is decoded in the CTEInfo bit field of the received frame,
- in AoA transmission, the transmitter does not need this information as it simply sends the CTE sequence on its unique antenna.

Bit 0 **AOD_NAOA**: indicates to the IP_BLE the type of CTE for transmission mode to manage or not an antenna switching sequence.

0: Angle of Arrival (AoA) type is used for the transmission

1: Angle of Departure (AoD) type is used for the transmission

This bit field is used only when StatMach.TxMode=1, TxRxPack.CTEAndSamplingEnable=1 and StatMach.CTEDisable=0.

25.11.22 WORD15 register (WORD15)

Address offset: 0x0054

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD15 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IQSAMPLESPTR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IQSAMPLESPTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IQSAMPLESPTR[31:0]**: Pointer to IQ samples storage buffer (received during CTE reception).

This pointer defines the start address of the RAM location where to store the received IQ samples during a Constant Tone Extension phase.

The IQ samples are stored in words built with 16-bit LSB for Q[15:0] samples and 16-bit MSB for I[15:0].

This bit field is used and verified only when StatMach.TxMode=0, StatMach.CTEDisable=0, TxRxPack.CTEAndSamplingEnable=1 and StatMach.MaximumIQSamplesNumber>0.

Note: this pointer is an absolute address.

Caution: this pointer address must be 32-bit aligned else the sequence is aborted at the end of the 1st INIT and STATUSREG.ADDPOINTERROR flag is raised.

25.11.23 WORD16 register (WORD16)

Address offset: 0x0058

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD16 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ANTENNAPATTERNPTR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ANTENNAPATTERNPTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ANTENNAPATTERNPTR[31:0]**: Pointer to Antenna Pattern (for antenna switching sequence).

This pointer defines the start address of the RAM location where to get the Antenna ID pattern to switch antennae during a Constant Tone Extension phase.

Then Antenna pattern is a list of 8-bit Antenna Identifiers.

The RAM buffer addressed by this pointer must contain at least StatMach.AntennaPatternLength bytes.

This bit field is used and verified only when TxRxPack.CTEAndSamplingEnable=1, StatMach.CTEDisable=0 and StatMach.MaximumIQSamplesNumber>0.

Note: this pointer is an absolute address.

Caution: this pointer address must be 32-bit aligned else the sequence is aborted at the end of the 1st INIT and STATUSREG.ADDPOINTERROR flag is raised.

PaPower bit field description

Table 136 provides the PA power correspondence to program the StatMach.PaPower bit field.

The SMPS of the SoC must provide a minimum voltage to reach the targeted PaPower:

SMPS output level = 1.4 V minimum up to 4 dBm

SMPS output level = 1.55 V minimum for 5 dBm

SMPS output level = 1.7 V minimum for 6 dBm

For 8 dBm, refer to the note after the table as this PaPower requests a specific configuration.

Table 136. PaPower bit field value and associated output power

Value	Output power (dBm)	Value	Output power (dBm)	Value)	Output power (dBm)	Value	Output power (dBm)
0x1F	+6 / +8 *	0x17	-0.5	0xF	-5.9	0x7	-14.1
0x1E	+5	0x16	-0.85	0xE	-6.9	0x6	-15.25
0x1D	+4	0x15	-1.3	0xD	-7.8	0x5	-16.5
0x1C	+3	0x14	-1.8	0xC	-8.85	0x4	-17.6
0x1B	+2	0x13	-2.45	0xB	-9.9	0x3	-18.85
0x1A	+1	0x12	-3.15	0xA	-10.9	0x2	-19.75
0x19	0	0x11	-4.00	0x9	-12.05	0x1	-20.85
0x18	-0.15	0x10	-4.95	0x8	-13.15	0x0	-40

Several settings are needed to reach the +8 dBm in transmission:

- program the SMPS located in the SoC to provide 1.9 V
- program 0x1F in StatMach.PaPower[4:0] bit field
- program 1 in StatMach.TxHp bit (to configure the LDO_TRANSFO in bypass mode).

TxRxPack RAM table

Table 137. TxRxPack RAM table(1)

Word	Byte addr	R/W by IP_BLE	7	6	5	4	3	2	1	0
0x00	0x00	R	NextPtr[7:0]							
	0x01		NextPtr[15:8]							
	0x02		NextPtr[23:16]							
	0x03		NextPtr[31:24]							
0x01	0x04		IncChan	SN_EN	Advertise	CrcInitSel	CTEAndSamplingEnable	KeepSemaReq	ChanAlgo2Sel	CalReq
	0x05		-	-	subEventChanAlgo2	DisableWhitening(2)	-	TxdataReady	AllTableReady	NextTxMode
	0x06	-	-							
	0x07	-	-							
0x02	0x08	R/W	DataPtr[7:0]							
	0x09		DataPtr[15:8]							
	0x0A		DataPtr[23:16]							
	0x0B		DataPtr[31:24]							
0x03	0x0C	R	timer2[7:0]							
	0x0D		timer2[15:8]							
	0x0E		TrigDone	TrigRcv	-	Timer2En	timer2[19:16]			
	0x0F		IntRcvOk	IntRcvCrcErr	IntTimeCapture	IntRcvCmd	IntRcvNoMd	IntRcvTimeout	IntDone	IntTxOk

1. Unused cells are filled with “-”.
2. Content of pink-shaded cells is for debug and qualification purposes.

25.11.24 TxRxPack registers

Table 138. TxRxPack register list

Offset	Register Name	Register Title	Reset
0x0000	WORD0	WORD0 register	0x0000 0000
0x0004	WORD1	WORD1 register	0x0000 0000
0x0008	WORD2	WORD2 register	0x0000 0000
0x000C	WORD3	WORD3 register	0x0000 0000

25.11.25 WORD0 register (WORD0)

Address offset: 0x0000

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD0 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NEXTPTR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEXTPTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NEXTPTR[31:0]**: Next pointer address entry of the linked list.
Points to the next transmit or receive packet. The user must enter the absolute address, not an offset.
Caution: This pointer must be 32-bit aligned else STATUSREG.ADDPOINTERROR is set (and INTERRUPT1REG.ADDPOINTERROR if GlobalStatMach.IntAddPointError=1).

25.11.26 WORD1 register (WORD1)

Address offset: 0x0004

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SUBEVENTCHANALGO2	DISABLEWHITENING	Res.	TXDATAREADY	ALLTABLEREADY	NEXTTXMODE	INCCHAN	SN_EN	ADVERTISE	CRCINITSEL	CTEANDSAMPLINGENABLE	KEEPSEMAREQ	CHANALGO2SEL	CALREQ
		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SUBEVENTCHANALGO2**: Select the SubEvent Channel computation in the channel incrementer block when the algorithm

Bit 12 **DISABLEWHITENING**: Whitening Disable

0: The whitening is enabled in the transmit block and in the receive block.

1: The whitening is disabled in the transmit block and in the receive block. This may be used for debug or during official Bluetooth compliance test.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TXDATAREADY**: Transmission data ready.

This bit is checked only if the current transfer is a transmission.

The check is done at the beginning of the DATA INIT phase to ensure that at least few bytes of the transmission payload are already written in the data buffer.

This bit allows doing an "On the fly" data buffer memcopy while transmission has already started on the antenna.

0: The transmission payload is no ready. The transfer is not started by the sequencer.

1: The transmission payload is ready, so the transfer is started by the sequencer.

Note: recommendation for Transmission data payload is to set this TxDataReady bit only when at least 16 bytes of data are available in the payload data buffer.

Bit 9 **ALLTABLEREADY**: All table data ready.

This bit is checked at the beginning of the 2nd INIT phase to ensure bit fields related to on-going transfer and about to be read are relevant.

0: The RAM table information related to the on-going transfer are not ready. The transmission is not started by the sequencer.

1: The RAM table information related to the on-going transfer are ready. The transmission is started by the sequencer.

Note: goal of this bit is to allow the software blocking a transfer if RAM table update is not over.

Bit 8 **NEXTTXMODE**: Flag indicating if next TxRx packet to be handled by the link controller StateMach is a receive packet or a transmit packet.

The sequencer overloads StateMach.TxMode value with NextTxMode value during each RAM write back phase.

0: next TxRx packet is a receive packet.

1: next TxRx packet is a transmit packet.

Bit 7 **INCCHAN**: Automatic channel incrementer enable.

When enabled, the automatic channel incrementer takes as input StateMach.UChan, TxRxPack.Advertise, TxRxPack.ChanAlgo2Sel, StateMach.Remap_chan, StateMach.hopincr, StateMach.UsedChannelFlags, StateMach.EventCounter and TxRxPack.SubEventChanAlgo2.

0: automatic channel incrementer is disabled

1: automatic channel increment is enabled.

See Channel number management chapter in UM for details.

Bit 6 **SN_EN**: Automatic SN, NESN hardware mechanism enable.

0: Automatic SN/NESN hardware mechanism is disabled. The receive pointers and transmit pointers are systematically shifted independently of SN, NESN bits and also on a receive timeout sequence.

1: Automatic SN/NESN hardware mechanism is enabled.

Bit 5 **ADVERTISE**: Advertise packet format

0: The packet format stored in RAM or to be received is a data packet format.

1: The packet format stored in RAM or to be received is an advertise packet format.

Bit 4 **CRCINITSEL**: CRC initialization value selector.

0: the transmit and the receive block initialize their CRC with a constant equal to: 0x555555

1: the transmit and the receive block initialize their CRC with the value defined by StateMach.CrcInit

Bit 3 **CTEANDSAMPLINGENABLE**: indicates to handle the Constant Tone Extension for this packet.

In transmission:

0: the IP_BLE does not append any CTE sequence at the end of the packet

1: the IP_BLE appends a CTE sequence at the end of the packet

In reception:

0: the IP_BLE manages the CTE detection only to extract the CTETime information, keeping reception active until the end of the CTE phase but does not manage any other features like tie slot sampling or potential antenna switching. The goal is to keep the coherency about “last bit on the air time stamp”for TIFS management and no more.

1: the IP_BLE manages the CTE detection and reacts accordingly to information extracted from the received frame to manage sampling time slots and potential antenna switching.

Bit 2 **KEEPSEMAREQ**: Request to keep the RRM semaphore.

Indicate if the IP_BLE needs to keep the RRM token at the end of the current transfer.

0: The token request is cleared when the controller starts its context saving.

1: The token request is maintained high at the end of the sequence.

Caution: This bit MUST be set to fit the IFS = 150 µs constraint.

Indeed, when the token is released, the Radio FSM switches back in IDLE mode. The Radio FSM needs around 45 µs more (ENA_RF_REG and ENA_CUR states) to go back to ACTIVE2 state on next Bluetooth LE sequence trig event.

Bit 1 **CHANALGO2SEL**: Channel hoping algorithm selection.

if TxRxPack.incchan = 0, this bit field has no effect.

if TxRxPack.incchan = 1:

0: The algorithm

Bit 0 **CALREQ**: Calibration request.

0: The RF PLL calibration is disabled. This setting is used when this calibration has already been done and if the radio did not go in low power state.

1: The calibration of the RF PLL is enabled. It must be performed at each channel frequency change or after wakeup.

25.11.27 WORD2 register (WORD2)

Address offset: 0x0008

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD2 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAPTR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAPTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATAPTR[31:0]**: Data pointer address.

Points to the data packet linked with TxRxPack (called DataPack in this document).

This data packet contains the header and the data, excluding the preamble, the access address and the CRC.

The Bluetooth LE link layer writes this packet in RAM in case of reception and reads it from RAM in case of transmission.

Note: this pointer has no memory address alignment requirement.

However the software must write an absolute address (not an offset). If the 8-bit MSB part of the pointer value is not equal to the RAM 8-bit MSB address, an ADDPOINTERROR flag is raised.

25.11.28 WORD3 register (WORD3)

Address offset: 0x000C

Reset value: 0x0000 0000 (0xFFFF FFFF)

WORD3 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTRCVOK	INTRCVCRCERR	INTTIMECAPTURE	INTRCVCMD	INTRCVNOMD	INTRCVTIMEOUT	INTDONE	INTTXOK	TRIGDONE	TRIGRCV	Res.	TIMER2EN	TIMER2[19:16]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **INTRCVOK**: Receive OK interrupt enable
0: The interrupt INTERRUPT1REG.RCVOK is disabled
1: The interrupt INTERRUPT1REG.RCVOK is enabled

Bit 30 **INTRCVCRCERR**: Receive CRC error interrupt enable
0: The interrupt INTERRUPT1REG.RCVCRCERR is disabled
1: The interrupt INTERRUPT1REG.RCVCRCERR is enabled

Bit 29 **INTTIMECAPTURE**: “Time Capture occurred”interrupt enable
0: The interrupt INTERRUPT1REG.INTTIMECAPTURETRIG is disabled
1: The interrupt INTERRUPT1REG.INTTIMECAPTURETRIG is enabled
Note: The event(s) responsible of the interrupt can be the sequencer Time Capture and/or the TrigDone and/or the TrigRcv events.

Bit 28 **INTRCVCMD**: Received packet is a command”interrupt enable
0: The interrupt INTERRUPT1REG.RCVCMD is disabled
1: The interrupt INTERRUPT1REG.RCVCMD is enabled

Bit 27 **INTRCVNOMD**: No more Data (end of connection found) interrupt enable
0: The interrupt INTERRUPT1REG.RCVNOMD is disabled
1: The interrupt INTERRUPT1REG.RCVNOMD is enabled

Bit 26 **INTRCVTIMEOUT**: Receive timeout interrupt enable
0: The interrupt INTERRUPT1REG.RCVTIMEOUT is disabled
1: The interrupt INTERRUPT1REG.RCVTIMEOUT is enabled

Bit 25 **INTDONE**: Done interrupt enable
0: The interrupt INTERRUPT1REG.DONE is disabled
1: The interrupt INTERRUPT1REG.DONE is enabled

Bit 24 **INTTXOK**: Interrupt enable of "good reception of transmitted packet is confirmed by the peer device".
0: The interrupt INTERRUPT1REG.TXOK is disabled
1: The interrupt INTERRUPT1REG.TXOK is enabled
Note: this interrupt must be enabled in the RxPack table as the feature is active at the end of a reception.

Bit 23 **TRIGDONE**: Time capture enable on "On air" last transmitted/received bit.

0: No time stamping in TIMERCAPTUREREG is achieved, no interrupt is generated by TrigDone.

1: The interpolated absolute time is captured in TIMERCAPTUREREG when the demodulator receives the last bit of the bit stream or when the last transmitted has been shift out of the transmit block.

When this bit is set and if a time capture event occurs, the STATUSREG.TIMECAPTURETRIG is set to 1. An interrupt is raised if enabled (associated to INTERRUPT1REG.TIMECAPTURETRIG set to 1).

Note: If GlobalStatMach.TimeCapture or TxRxPack.TrigRcv bit is set, the TIMERCAPTUREREG IP_BLE APB register show this last event trigger value at the end.

Bit 22 **TRIGRCV**: Time capture enable on received preamble and access address pattern detection.

0: No time stamping requested on preamble + access address detection.

1: The interpolated absolute time is captured in TIMERCAPTUREREG when the demodulator detects the preamble + access address in the received bit stream.

When this bit is set and if a time capture occurs, the STATUSREG.TIMECAPTURETRIG is set to 1. An interrupt is raised if enabled (associated to INTERRUPT1REG.TIMECAPTURETRIG set to 1).

This bit must be set to 0 in transmission TxRxPack table not to disturb other time capture options.

Note: If GlobalStatMach.TimeCapture or TxRxPack.TrigDone bit is set, the TIMERCAPTUREREG IP_BLE APB register shows this last event trigger value at the end.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TIMER2EN**: Timer2 enable (for next timer trig).

0: Timer2 is not enabled at the end of this current packet.

1: Timer2 is enabled at the end of this current packet.

Bits 19:0 **TIMER2[19:0]**: Timer2 triggering value setting.

Defines the delay before next Timer2 trigger event if TxRxPack.Timer2En=1.

Time unit is in microseconds.

Note: the Timer2 delay starts a bit earlier than the end of the on-going sequence (on last transmitted bit or last received bit and before the context saving phase).

DataPack RAM table

The DataPack tables are the data buffer for reception or transmission packet. They are pointed by the TxRxPack.DataPtr value.

Their content corresponds to the PDU (header bytes, payload and potentially MIC for encrypted packets).

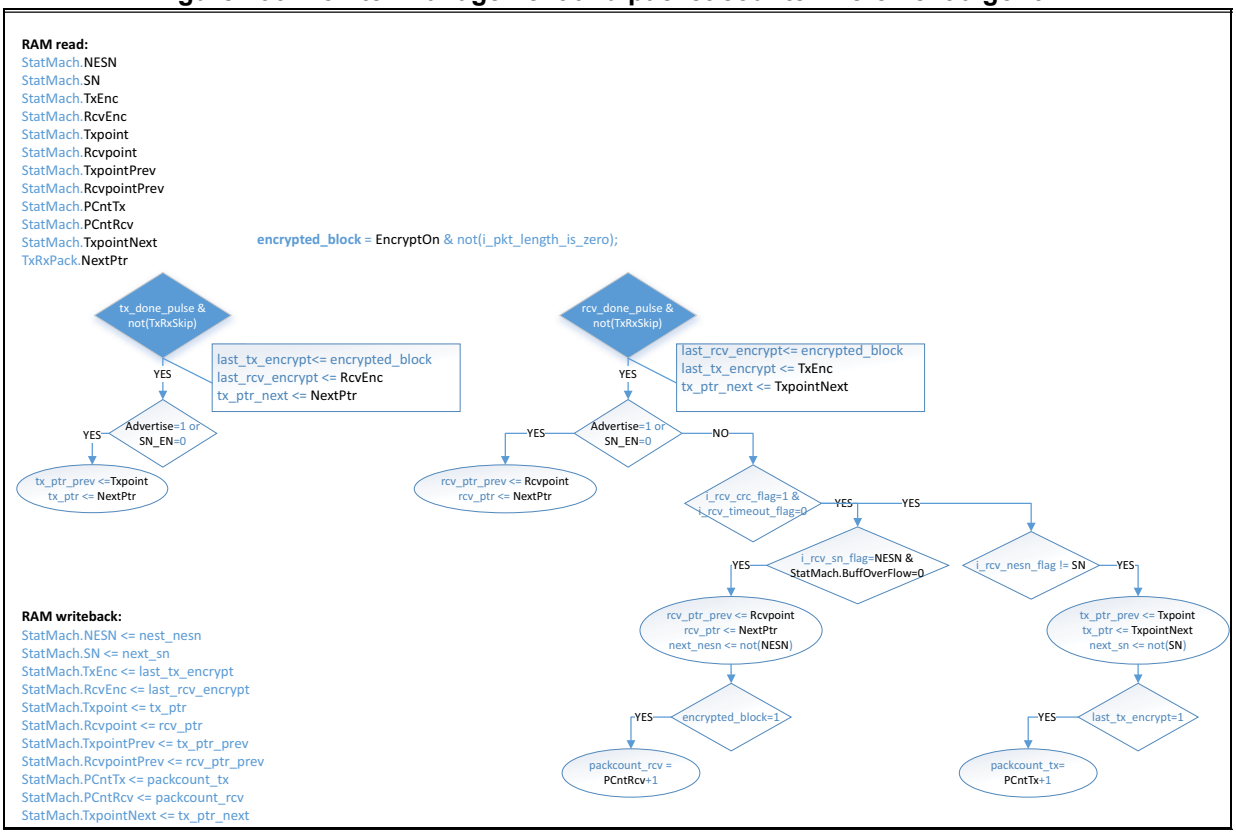
25.11.29 Complementary information

Pointer management and packet counter

The sequencer updates the pointers and packet counters at the end of a transmission or a reception, depending on some parameters.

The *Figure 206* illustrates the actions done on the pointers and packet counter.

Figure 206. Pointer management and packet counter increment algorithm



Channel number management

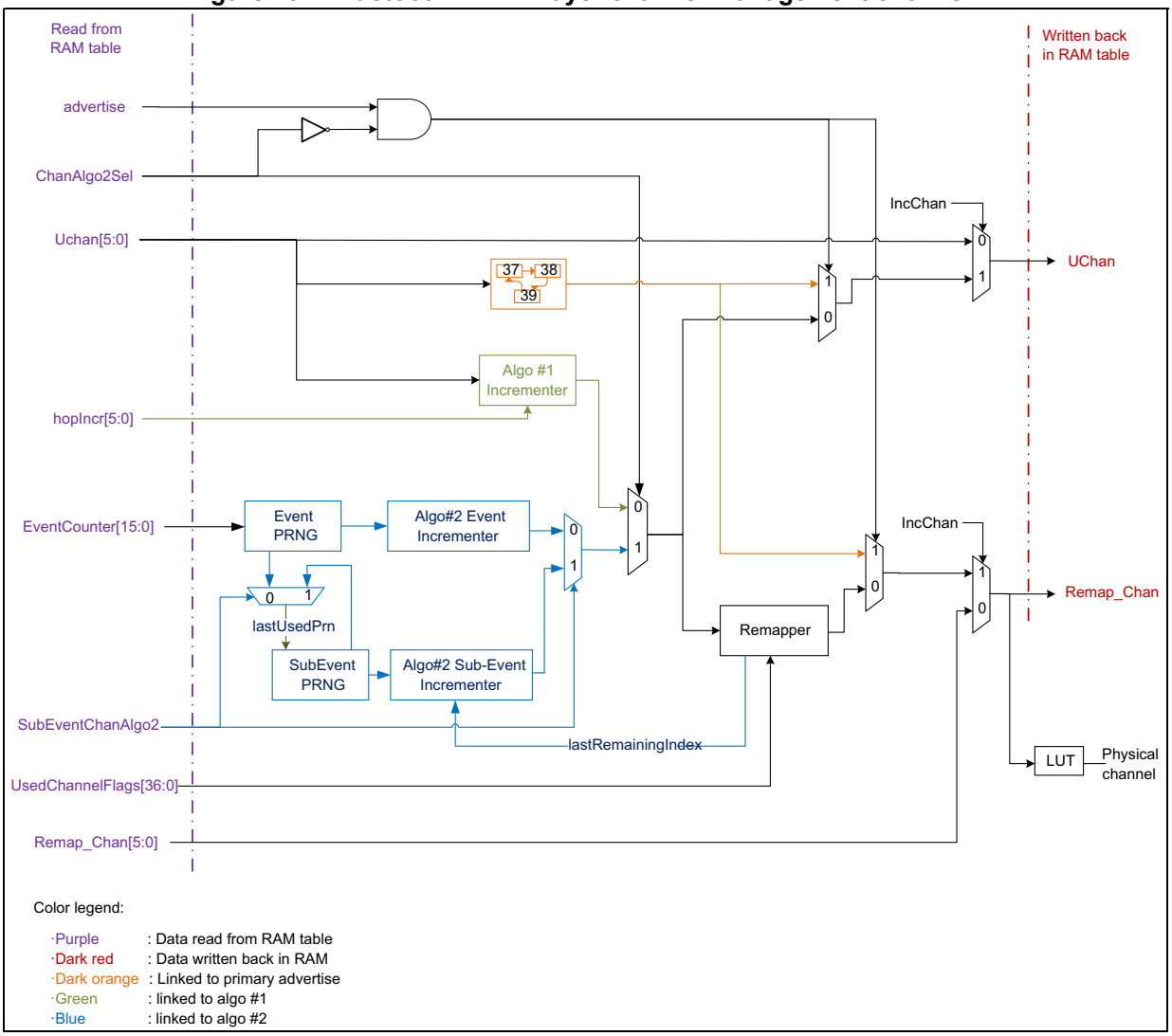
The Bluetooth LE link layer manages the channel frequency through different parameters located in the RAM table.

A channel incrementer allows calculating a new channel if TxRxPack.IncChan bit is set. The algorithm (#1 or #2) is selected through the TxRxPack.ChanAlgo2Sel bit.

In addition, a remap is done to fit with the StatMach.UserChannelFlags if TxRxPack.IncChan bit is set.

The *Figure 207* presents an overview of the channel number management.

Figure 207. Bluetooth LE link layer channel management overview



When the channel incrementer HW block is used (IncChan = 1), the selection of the algorithm must be managed by the SW following the truth table presented in *Table 139*.

Table 139. Truth table to select the correct algorithm

TxRxPack			Selected algorithm
Advertise	ChanAlgo2Sel	SubEventChanAlgo2	
0	0	X	Algorithm #1 is used for the channel hopping for data channel.
1	0	X	Primary advertising channels selected with automatic incrementation as follows: 37 → 38 → 39 → 37 → 38... Note: this configuration should not be used when running on secondary advertising channels (undefined behavior).
0	1	0	Algorithm #2 for an event channel (connection, secondary Advertising or CIS/BIS event) is used for the channel hoping for data channel.
0	1	1	Algorithm #2 for a subevent channel (e.g. CIS/BIS subevent) is used for the channel hoping for data channel.
1	1	0	Algorithm #2 is used for the channel hoping for Periodic Advertising channel. Note: this configuration should not be used when running on primary advertising channels (undefined behavior).
1	1	1	Not supposed to be used with Bluetooth specification/use-cases.

In addition, the following table lists the different bit field in RAM tables linked to the channel number management and indicates which ones are used at HW level according to targeted algorithm (#1 or #2).

Table 140. RAM table bit fields usage versus algorithm number

Bit field	RAM Table	Read/Write-back	Considered	
			for Algo #1	for Algo #2
InChan	TxRxPack	Read	Yes	Yes
ChannelAlgo2Sel	TxRxPack	Read	Yes	Yes
EventCounter[15:0]	StatMach	Read	No	Yes
SubEventChanAlgo2	TxRxPack	Read	No	Yes
advertise	TxRxPack	Read	Yes	Yes
accaddr[31:0]	StatMach	Read	No	Yes
UsedChannelFlags[36:0]	StatMach	Read	Yes	Yes
hopincr[5:0]	StatMach	Read	Yes	No
UChan[5:0]	StatMach	Read / Written back	Yes	No
Remap_chan[5:0]	StatMach	Read / Written back	Yes	Yes

Time capture

The IP_BLE can capture the absolute time on specific events.

The capture feature is enabled inside the RAM tables and result is provided in a IP_BLE APB register called TIMECAPTUREREG[31:0].

The events that can be time stamped / captured are:

- the sequencer reaches the end of 1st INIT step (***InitDelay*** timeout)
- the sequencer reaches the end of DATA INIT step (init_radio_delay timeout)
- “On the air” last bit transmitted/received bit (depending on RX or TX current mode)
- Preamble + Access Address detection event on a reception

The time capture service is associated to an interrupt flag that is enabled through the TxRxPack.IntTimeCapture bit and status / interrupt flag at the end of the sequence is available in the IP_BLE APB STATUSREG.TIMECAPTURETRIG (and INTERRUPT1REG.TIMECAPTURETRIG if interrupt is enabled inside the TxRxPack table).

If several events are requested to be captured on a same sequence (for example, the sequencer reaches end of 1st INIT and “on the air” last bit, the latest occurrence is the available time inside the TIMECAPTUREREG register.

On a transmission sequence:

Table 141. Time capture parameters on transmission sequence

GlobStatMach		TxRxPack		APB IP_BLE
TimeCapture	**TimeCaptureSel**	**TrigRcv**	**TrigDone**	**TIMECAPTUREREG**
0	x	x	0	Sequencer end of 1st INIT
1	0	x	0	Sequencer end of 1st INIT
1	1	x	0	Sequencer end of DATA INIT
x	x	x	1	“On the air” last transmitted bit

On a reception sequence:

Table 142. Time capture parameters on transmission sequence

GlobStatMach		TxRxPack		APB IP_BLE
TimeCapture	**TimeCaptureSel**	**TrigRcv**	**TrigDone**	**TIMECAPTUREREG**
0	x	0	0	Sequencer end of 1st INIT
1	0	0	0	Sequencer end of 1st INIT
1	1	0	0	Sequencer end of DATA INIT
x	x	1	0	Preamble + Access Address detection time
x	x	x	1	“On the air” last received bit

Sequencer timings recommended values

The 2nd INIT and DATA INIT steps of the sequencer use several timing parameters in addition to the init_radio_delay. This generic name covers 4 different durations depending on the transfer configuration (Tx/RX, PLL calibration/No calibration).

Reminder: the Radio FSM and the sequencer run in parallel with almost no handshake from the start of the 2nd INIT phase.

A set of timings is programmed in the GlobalStatMach to guarantee that the sequencer starts the transmission (respectively reception) phase with respect of the Radio FSM progress.

Figure 208 and *Figure 209* summarize the timings involved.

Figure 208. Timings of a RX sequence

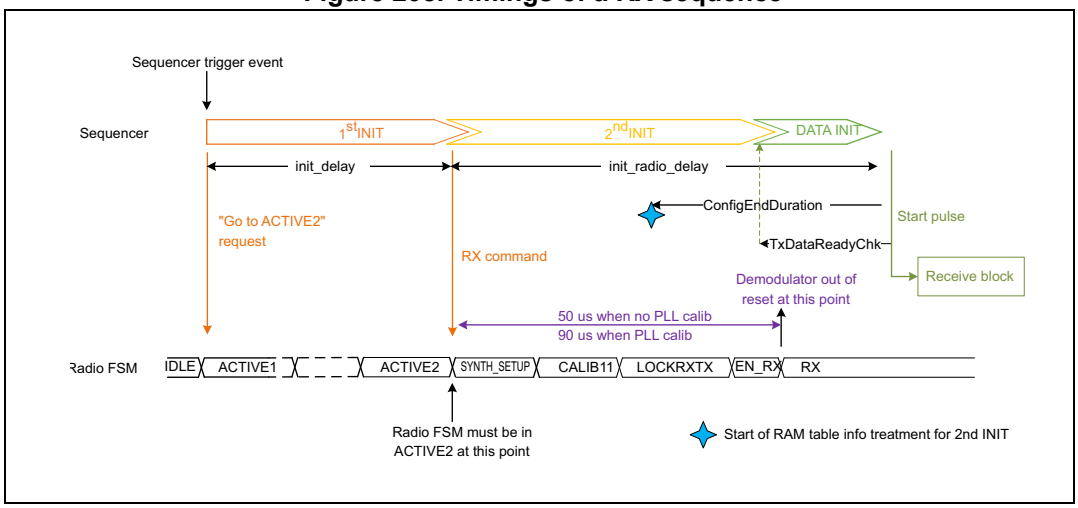
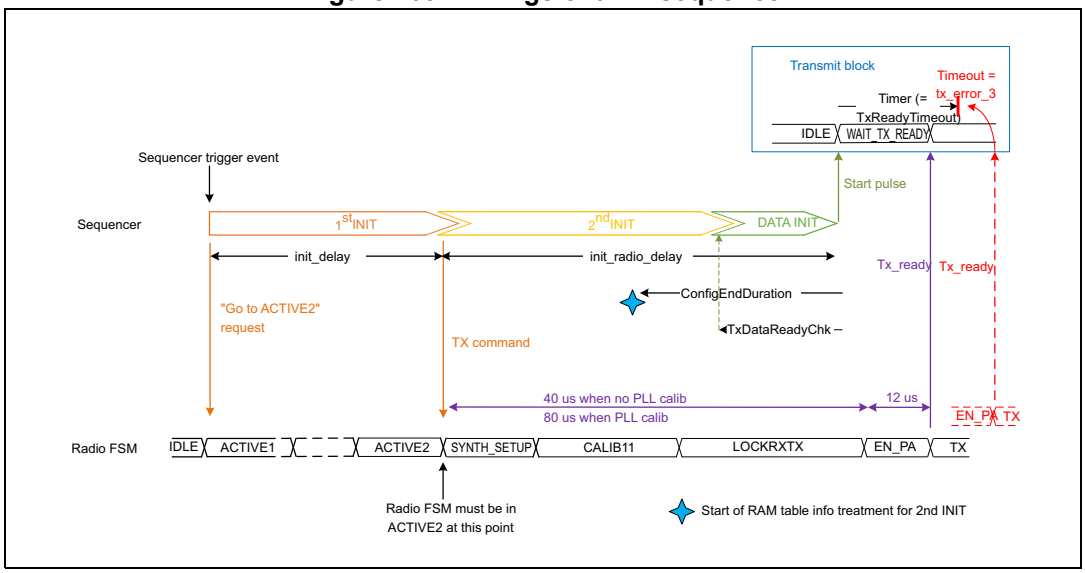


Figure 209. Timings of a TX sequence



Reminder on main constraints for a transmission:

- the DATA_INIT step must end before the Radio FSM reaches the TX state
  - it is possible to take some margin as the TxReadyTimeout delay allows waiting the confirmation the Radio FSM is in TX state before to really start the TxDelayStart timeout (see *Figure 203: TX sequence*).
- If TxRxPack.IncChan = 0, the sequencer read the PaPower[4:0] information from RAM table only after the ConfigEndDuration temporization.
  - the ConfigEndDuration value must be big enough to ensure the sequencer treat this information before the Radio FSM reaches the EN_PA state.

Reminder on main constraints for a reception:

- the DATA_INIT step must end close to the moment the Radio FSM reaches the RX state as it corresponds to the start of the Receive timeout window.
- The sequencer transfers the StatMach.accaddr information in the AA0_DIG_USR to AA3_DIG_USR Radio registers for the demodulator block. This transfer in Radio register is done only after the ConfigEndDuration temporization. This information is used by the demodulator and must be present before the Radio FSM reaches the RX state:
  - the ConfigEndDuration value must be big enough to ensure the sequencer starts the activity before the Radio FSM reaches the RX state.

Table 143 shows some recommendations / proposals on values for some timings programmed through RAM tables.

Table 143. Delays for sequencer 2nd INIT step proposal

GlobalStatMach table bit field	Recommended value	Comments
TransmitNoCalDelayChk[7:0]	0x32 (50d)	Theoretical exact duration from ACTIVE2 to TX state is 52 µs. Keep some margin to avoid timeout after TX state. Then, use the TxReadyTimeout[7:0] to define the duration of the wait TX state info for the transmit block.
TransmitCalDelayChk[7:0]	0x5A (90d)	Theoretical exact duration from ACTIVE2 to TX state is 92 µs. Keep some margin to avoid timeout after TX state. Then, use the TxReadyTimeout[7:0] to define the duration of the wait TX state info for the transmit block.
ReceiveNoCalDelayChk[7:0]	0x32 (50d)	Theoretical exact duration from ACTIVE2 to RX state is 50 µs. The delay can be programmed to 50 µs.
ReceiveCalDelayChk[7:0]	0x5A (90d)	Theoretical exact duration from ACTIVE2 to RX state is 90 µs. The delay can be programmed to 90 µs.
ConfigEndDuration[7:0]	at least 10 max is init_radio_delay	Note: if this value is too small, the sequencer may not do all the AHB RAM table accesses on time. the user must take margin (not use the minimum value) when a concurrent AES operation occurs during the 2nd INIT of the sequencer (Manual AES or LE Privacy)

Table 143. Delays for sequencer 2nd INIT step proposal (continued)

GlobalStatMach table bit field	Recommended value	Comments
TxDataReadyCheck[7:0]k	between 1 and 5	Can be increased to take more margin. Only impact is a potential useless extra waiting time before to abort when the Radio FSM / the sequence is broken.
TxReadyTimeout[7:0]	at least 5	Can be increased to take more margin. Only impact is the TX sequence abort is delayed with the same additional delay in case of real Radio FSM TX sequence issue.

25.11.30 Angle of arrival (AoA) and angle of departure (AoD)

The Bluetooth Core Specification v5.1 introduced new capabilities that support higher-accuracy direction finding.

The Bluetooth direction finding exploits some of the fundamental properties of radio waves by taking several phase and amplitude measurements (across different antenna) that can be used in direction finding calculations at precise intervals in a process known as In-phase and Quadrature Sampling (IQ sampling).

Applications use this data in calculations that involve trigonometry and information about the design of the antenna array.

A single IQ sample consists of the wave's amplitude and phase angle represented as a set of Cartesian coordinates. Applications can transform this Cartesian representation into corresponding polar coordinates that yield the phase angle and the amplitude value.

Refer to *[2]* for details on this new feature.

The existing RAM tables have been upgraded to support the new HW features (CTE, I/Q sampling and Antenna Switching) added to support the direction-finding topic.

RAM tables and registers impact

The existing RAM tables have been upgraded to support the new HW features (CTE, I/Q sampling and Antenna Switching) added to support the direction-finding topic.

GlobalStatMach RAM table

The GlobalStatMach table size has not been impacted: still 7 words = 28 bytes.

A new bit field has been added in the RFU existing Word6: DefaultAntennaId[6:0] bit field in Word6[6:0].

This bit field is mandatory for the configurations where more than one antenna is present on the board.

This bit field indicates which antenna identifier must be used to transmit or receive packets without CTE or packet body (Preamble, Access Address, PDU and CRC) for CTE enabled packets. This value is transmitted through the corresponding pads of the device to the external component located on the board to switch on the requested antenna.

StatMach RAM table

The StatMach RAM table size has been increased of 3 words for a total of 23 words = 92 bytes.

The additional bit fields are the following:

- CTEDisable bit in Word0[27]. If set,
  - in transmission: no CTE appended at the end of the Frame whatever the **TxRxPack.CTEAndSamplingEnable** bit value
  - in reception: no CTE detection nor treatment done on the PDU, frame treated as a Bluetooth core specification 5.0, or prior, format.
- CTETime[4:0] bit field in Word14[6:2]:
  - used only when **StatMach.CTEDisable=0, TxRxPack.CTEAndSamplingEnable=1 and StatMach.TxMode=1,**
  - same format as the CTEInfo CTETime bit field (time unit = 8 µs),
  - provides the duration of the Constant Tone Extension to append after CRC.
- CTESlotWidth bit in Word14[1]
  - in transmission and AoD use-case: used to control the antenna switching timing,
  - in reception and in AoA use-case: used to control the antenna switching and the I/Q sampling timings,
  - not needed in other configurations,
  - used only when **StatMach.CTEDisable=0, TxRxPack.CTEAndSamplingEnable=1**
- MaximumIQSamplesNumber[6:0] bit field in Word14[14:8]
  - needed in reception only to indicate the maximum I/Q samples that can be stored in RAM,
  - if more I/Q samples are received (CTE Time longer than maximum samples allowed), the sequence goes on except the AHB master stops storing the additional receive samples.
  - used only when **StatMach.CTEDisable=0, TxRxPack.CTEAndSamplingEnable=1 and StatMach.TxMode=0**
- IQSamplesPtr[31:0] bit field in Word15[31:0]
  - needed in reception only to indicate the start address in RAM to store the I/Q samples received during the CTE,
  - the address contained by this pointer must be 32-bit aligned and MSByte value must be equal to the MSByte of the RAM base address of the device,
  - used and verified only when **StatMach.CTEDisable=0, TxRxPack.CTEAndSamplingEnable=1, StatMach.TxMode=0** and **StatMach.MaximumIQsamplesNumber >0**.
- AntennaPatternLength[7:0] bit field in Word14[23:16]
  - needed when antenna switching must be managed on the current transfer (in transmission if AoD / in reception if AoA),
  - defines the length of the Antenna ID pattern to be applied during the CTE. If the CTE duration requires more slots than the length defined here, the same pattern is repeated until the end of the CTE phase,
  - used only when **StatMach.CTEDisable=0 and TxRxPack.CTEAndSamplingEnable=1.**

- AoD_nAoA bit in Word14[0]
  - Used only when **StatMach.CTEDisable=0, TxRxPack.CTEAndSamplingEnable=1** and **StatMach.TxMode=1**,
  - indicates to the transmitter if it has to manage antenna switching (AoD) or not (AoA) during the transmission when CTE is enabled.
- AntennaPatternPtr[31:0] bit field in Word16[31:0]
  - needed when antenna switching has to be managed to indicate the start address of the Antenna Pattern (the antenna pattern is a list of 8-bit values describing the antenna identifier),
  - the RAM location addressed by the **StatMach.AntennaPatternPtr** shall contain at least **StatMach.AntennaPatternLength** bytes
  - used and verified only when **StatMach.CTEDisable=0, TxRxPack.CTEAndSamplingEnable=1** and **StatMach.AntennaPatternLength>0**.

TxRxPack RAM table

The TxRxPack RAM table size has been decreased of 1 word for a total of 4 words = 16 bytes.

A new CTEAndSamplingEnable bit has been added in Word1[3]. The role of this bit is to indicate to the Radio controller MR_BLE:

- In transmission:
  - to append a Constant Tone Extension after the CRC,
  - to manage the antenna switching if AoD use-case
- in reception:
  - to detect if the received frame has a CTE and to extract details from the CTEInfo,
  - to manage the I/Q Sampling storage in RAM,
  - to manage the antenna switching if AoA use-case.

Manage the feature in transmission

If the SW wants to append a Constant Tone Extension at the end of the frame, it has to:

- ensure the **StatMach.CTEDisable** bit is 0,
- program the CTE duration in the **StatMach.CTETime[4:0]** (same format as the CTETime bit field of the CTEInfo defined by the standard),
- set the **TxRxPack.CTEAndSamplingEnable** = 1 for this dedicated packet.

In the case of an encrypted frame, the AES encrypts (or not) the HEADER3 according to the TxRxPack.Advertise information.

Note: The transmit block does not decode on the fly the bytes provided by the RAM Data Buffer (located at DataPtr address) to identify the CTE information. It relies on the information provided through the RAM table. It is the responsibility of the SW to guarantee the coherency between the CTE related bit field in RAM and the data in RAM that corresponds to the CTEInfo bit field inside the frame on the air.

In addition to the CTE append on the transmitted frame, if the device must manage the antenna switching (AoD configuration), the SW has to:

- define the default Antenna ID for the packet body through the **GlobalStatMach.DefaultAntennaId[6:0]** bit field,
- provide the start address of the antenna pattern through the **StatMach.AntennaPatternPtr[31:0]** bit field,
- provide the antenna pattern / sequence length through the **StatMach.AntennaPatternLength[7:0]** bit field,
- fill the RAM location pointed by the **StatMach.AntennaPatternPtr[31:0]** bit field with 8-bit antenna identifier (the SW must ensure this RAM area contains at least **StatMach.AntennaPatternLength[7:0]** antenna ID),
- if needed, tune the TX_TIME_TO_SWITCH[6:0] timing in the ANTSW2_DIG_USR radio register to compensate potential delay between modulator to antenna path versus antenna switching component control signals.

In parallel, the SW must ensure the SoC GPIOs have been programmed in the accurate configuration to output the Antenna Identifier and enable signals to the external component.

To execute a transmission without CTE phase appended at the end of the frame, the SW just must set the **StatMach.CTEDisable** bit to 1.

Manage the feature in reception

CTE detection and decoding

If the SW wants the Radio controller MR_BLE to execute a reception with the ability to detect if the received frame is a packet with or without CTE, it has to:

- set the **StatMach.CTEDisable** bit to 0,
- set the **TxRxPack.CTEAndSamplingEnable** bit to 1.

In case of encrypted frame, the AES decrypts (or not the HEADER3 according to the TxRxPack.Advertise information.

Table 144 shows the behavior according to the combination between **TxRxPack.CTEAndSamplingEnable** and **StatMach.CTEDisable**.

Table 144. Behavior versus CTEDisable and CTEAndSampling bits value

Configuration	Behavior
StatMach.CTEDisable = 1 and TxRxPack.CTEAndSamplingEnable = “don’t care”	**The receiver behaves as a Bluetooth LE SIG 5.0 or before**. the receiver does not try to decode any CTEInfo bit field inside the received frame, the reception ends after CRC (if no DisableCrc bit set) the Timer2 (if enabled) starts at the end of the CRC.
StatMach.CTEDisable = 0 and TxRxPack.CTEAndSamplingEnable = 0	The receiver detects if CTE packet or not but does not sample the I/Q. This configuration allows keeping the synchronization on T_{IFS} with the transmitting device. the receiver decodes any CTEInfo bit field inside the received frame if present, the reception ends after the CRC (if not a CTE packet) or after the CTE phase (if CTE packet detected) the Timer2 (if enabled) starts at the end of the reception (after CRC or CTE).
StatMach.CTEDisable = 0 and TxRxPack.CTEAndSamplingEnable = 1	The receiver detects if CTE packet or not and manages the I/Q sampling. the receiver decodes any CTEInfo bit field inside the received frame if present, the reception ends after the CRC (if not a CTE packet) or after the CTE phase (if CTE packet detected) the Timer2 (if enabled) starts at the end of the reception (after CRC or CTE). the Radio controller MR_BLE stores the I/Q sampling in RAM (according to options programmed in the RAM tables by the SW)

During the reception, the receiver extracts from the frame the CTE info like duration, slot width and type.

The only configuration for which the Radio controller MR_BLE needs to get the information from the RAM table is the Angle of Arrival (AoA). In this specific case, the slot width is not indicated in the CTEType[1:0] bit field:

- 00: AoA
- 01: AoD with 1 µs slots
- 10: AoD with 2 µs slots
- 11: Reserved for future used

In AoA, the receiver device is the owner of the antenna switching and of the time slot width choice. In this specific configuration, the SW must provide the information through the **StatMach.CTESlotWidth** bit.

I/Q sampling

The Radio controller MR_BLE stores the I/Q samples in words built as follows

- I[15:0] as 16 MSbit
- Q[15:0] as 16 LSbit

As the AHB master writes both I and Q in one 32-bit access in RAM, the IQSamplesPtr must contain a 32-bit aligned address.

To manage a reception in CTE with I/Q sampling, the SW has:

- to set the reception has CTE aware (see *CTE detection and decoding*),
- to program the maximum number of I/Q samples that are allowed being written in RAM (between 0 and 127 I+Q) through the **StatMach.MaximumIQSamplesNumber[6:0]** bit field,
- to define the start address to store the I/Q samples in the **StatMach.IQSamplesPtr[31:0]**.

At the end of the reception, some status bit fields are provided to give visibility on what has been received:

- IP_BLE APB register called STATUS2REG [0] = IQSAMPLESREADY.
  - when set, indicates I/Q sampling has been received
  - when set, means that the received frame embedded a Constant Tone Extension at the end
- IP_BLE APB register called STATUS2REG [7:1] = IQSAMPLESNUMBER[6:0]
  - provides the number of I/Q samples stored in RAM

Antenna switching

In addition to the CTE detection and the I/Q sampling storage, in AoA, the device has to manage the antenna switching. To achieve this, the SW has to:

- define the default Antenna ID for the packet body through the **GlobalStatMach.DefaultAntennaId[6:0]** bit field,
- provide the start address of the antenna pattern through the **StatMach.AntennaPatternPtr[31:0]** bit field,
- provide the antenna pattern / sequence length through the **StatMach.AntennaPatternLength[7:0]** bit field,
- fill the RAM location pointed by the **StatMach.AntennaPatternPtr[31:0]** bit field with 8-bit antenna identifier (the SW must ensure this RAM area contains at least **StatMach.AntennaPatternLength[7:0]** antenna ID),
- if needed, tune the RX_TIME_TO_SWITCH[5:0] timing in the ANTSW1_DIG_USR radio register to compensate potential delay between antenna path to demodulator versus antenna switching component control signals,
- if needed, tune the RX_TIME_TO_SAMPLE[6:0] timing in ANTSW0_DIG_USR radio register to center the I/Q sampling action inside the slot window.

In parallel, the SW must ensure the SoC GPIOs have been programmed in the accurate configuration to output the Antenna Identifier and enable signals to the external component.

Note: The HW automatically restarts the antenna switching pattern from its first element if there are more CTE slots than the AntennaPatternLength value provided in the StatMach.

Error management

Several error cases are treated at HW level concerning CTE, I/Q sampling and antenna switching.

Invalid CTEType received

Expected values for the CTEType[1:0] bit field inside the CTEInfo are “00”, “01” or “10”. If the receiver decodes a CTEType[1:0] = “11” in a CTE packet, it behaves as if TxRxPack.CTEAndSamplingEnable = 0 (CTE time duration respected for reception phase but no I/Q sampling nor antenna switching management).

CTE length outside [20..2] window

The Radio controller MR_BLE IP has been designed to support CTE length outside the standard limit to offer possibility to develop proprietary version or for trials in lab.

In reception:

- if CTE length > 20:
  - the reception is able to manage I/Q sampling and CTE phase for CTE length greater than the standard. For I/Q sampling, the limit used by the HW is the StatMach.MaximumIQSamplesNumber bit field value.
- if CTE length < 2
  - the reception is able to manage I/Q sampling and CTE phase for CTE length less than the standard.

In transmission:

- if CTE length > 20:
  - the transmission is able to manage a CTE duration longer than the longest indicated by the Bluetooth core specification *[2]*.
  - the TXERROR_4 flag (and associated interrupt if enabled) is raised to inform the packet is not CTE standard compliant (but the transmission is executed with requested configuration).
- if CTE length < 2
  - the transmission is able to manage a CTE duration shorter than the shortest indicated by the Bluetooth core specification *[2]*. Note that for duration = 0, this is equivalent to no CTE phase after CRC.
  - the TXERROR_4 flag (and associated interrupt if enabled) is raised to inform the packet is not CTE standard compliant (but the transmission is executed with requested configuration).

CTE requested while Long Range is selected

The Bluetooth core specification *[2]* indicates that the Constant Tone Extension feature does not concern the Coded PHY packets (long range configuration).

If the Radio controller MR_BLE IP is in front of a contradictory configuration indicating coded PHY format and CTE enabled, the transfer is managed as a Coded PHY without CTE.

In reception, no CTE detection is activated (so no I/Q sampling nor antenna switching is managed neither).

In transmission:

- no CTE phase is appended at the end of the frame,
- the TXERROR_4 flag (and associated interrupt if enabled) is raised to information the configuration is not CTE standard compliant.

IQSamplesPtr[31:0] or AntennaPatternPtr[31:0] not 32-bit aligned

The StatMach.IQSamplesPtr[31:0] and the StatMach.AntennaPatternPtr[31:0] must contain a 32-bit aligned address.

For IQSamplesPtr, an error is detected if:

- the StatMach.IQSamplesPtr[31:0] does not contain a modulo 4 value (32-bit aligned address),
- and StatMach.CTEDisable=0,
- and StatMach.CTEAndSamplingEnable=1,
- and StatMach.MaximumIQSamplesNumber[6:0] > 0

For AntennaPatternPtr, an error is detected if:

- the StatMach. AntennaPatternPtr [31:0] does not contain a modulo 4 value (32-bit aligned address),
- and StatMach.CTEDisable=0,
- and StatMach.CTEAndSamplingEnable=1,
- and StatMach.AntennaPatternLength[7:0] > 0

The associated error is a ConfigError (no RF transfer as the sequencer aborts after the 1st INIT step). Refer to *Configuration error* for details on the ConfigError behavior.

IQSamplesPtr[31:24] or AntennaPatternPtr[31:24] not equal to device RAM base address[31:24]

The StatMach.IQSamplesPtr[31:24] and the StatMach.AntennaPatternPtr[31:24] must contain a value equal to the RAM base address[31:24].

For IQSamplesPtr, an error is identified if:

- the StatMach.IQSamplesPtr[31:24] is different from the RAM base address[31:24],
- and StatMach.CTEDisable=0,
- and StatMach.CTEAndSamplingEnable=1,
- and StatMach.MaximumIQSamplesNumber > 0

For AntennaPatternPtr, an error is detected if:

- the StatMach. AntennaPatternPtr [31:24] is different from the RAM base address[31:24],
- and StatMach.CTEDisable=0,
- and StatMach.CTEAndSamplingEnable=1,
- and StatMach.AntennaPatternLength[7:0] > 0

The associated error is an AddPointError (no RF transfer as the sequencer aborts after the 1st INIT step). Refer to *Address Pointer error* for details on the AddPointError behavior.

I/Q sampling storage overflow

In case of too high latency in AHB transfers between the Radio controller MR_BLE IP and the RAM inside the device, some received I/Q samples could be overwritten inside the internal buffer before being transferred into the SoC RAM.

In this case:

- a status flag is raised in the STATUS2REG APB register: STATUS2REG [29] = IQSAMPLESMISSINGERROR,
- no more I/Q samples are written in RAM until the end of the CTE phase,
- the IQSAMPLESREADY bit and the IQSAMPLESNUMBER bit fields are updated according to real number of samples written in RAM,
- the rest of the reception goes on in a normal way (including antenna switching is AoA, despite no more samples is recorded).

Then, the SW has the choice to handle or not the partial list of received I/Q samples.

Note: This error is not supposed to occur as an internal FIFO has been put in place to buffered some I/Q samples and is supposed to be dimensioned according to the device bandwidth.

Antenna pattern length = 1

Despite the Bluetooth core specification, the HCI_LE_Set_Connection_CTE_Receive_Parameters indicates the Length_of_switching_Pattern parameter must be at least 2, the Radio controller MR_BLE IP, at HW level, supports an antenna pattern length=1.

In this case, the reference period and the sample slots are done on the same antenna.

Antenna Pattern read underflow

In case of too high latency in AHB transfers between the Radio controller MR_BLE IP and the RAM inside the device, the next antenna ID may not be ready inside the antenna switching sub-block when the next antenna switching event is supposed to occur.

In this case:

- a flag is raised in the STATUS2REG APB register: STATUS2REG [30] = ANTENNASWICTHINGPATTERNACCESSERROR ,
- the antenna switching process goes on but with an unpredictable value on the antenna ID and shifted value once the read value is available

Note: This error is not supposed to occur as an internal FIFO is present to store some anticipated antenna ID and is supposed to be dimensioned according to the device bandwidth.

25.11.31 AES

The Radio controller MR_BLE IP embeds an AES hardware accelerator. This accelerator is encapsulated in a wrapper allowing to share a single accelerator core for 3 different actions/modes:

1. on-the fly encryption
2. manual encryption
3. LE Privacy

In case of simultaneous requests for several modes, a priority mechanism serves the requester in the same order as the list order above.

On-the-fly encryption

This mode corresponds to an on-going Bluetooth encrypted transmission/reception.

In transmission, the AES encrypts the data read from the RAM DataPack table before to transmit them.

In reception, the AES decrypts the received data before to store them in the DataPack RAM table.

This mode is activated as soon as a RF transfer is done with the StatMach.EncryptOn bit set to ‘1’.

The on-the-fly AES feature can discriminate the part of the frame not concerned by encryption like the HEADER parts (including the HEADER3 when CTE is active in the frame).

Warning: The Isochronous Channels encryption is not managed by this HW feature and need to be done at SW level without enabling the HW encryption.

This mode is the most priority mode and is always be served first in case of concurrent requests not to impact the Bluetooth communication.

Manual encryption

The goal of this mode is to share the AES core embedded with the CPU of the SoC. As the Bluetooth LE link layer does not use this HW resource all the time, the idea is to let the CPU process its own encryptions through this HW accelerator. It avoids adding an extra AES core in the SoC if the share usage is acceptable.

A set of registers is present to manage this manual encryption (see *Section 25.12: Bluetooth LE IP registers*).

The process to do a manual encryption is the following:

1. Enter the key in the MANAESxKEYREG registers (x from 0 to 3)
2. Enter the text to encrypt (16 bytes by computation) in the MANAESCLEARTEXTxREG (x from 0 to 3) registers.
3. Possibility to enable an interrupt to be informed when the computation is over by setting the MANAESCMDREG[1] = INTENA bit

4. Launch the encryption by setting the MANAESCMDREG[0] = START bit.
5. Wait for end of computation:
    a. If the interrupt mode is enabled, the interrupt line dedicated to AES (irq_BLE_int2 also called BLE_AES) is raised and the CPU gets the reason in the INTERRUPT2REG[0] = AESMANENDINT bit.
    b. else, the SW has to poll the MANAESSTATREG[0] = BUSY bit until it is cleared by HW.
6. If interrupt is used, write 1 in the INTERRUPT2REG[0] = AESMANENDINT bit to clear the flag.
7. The encrypted data / result is available in the MANAESCYPHERTEXTxREG (x from 0 to 3) registers.
8. If needed the SW can restart from step 1 or step 2.

LE privacy

In Radio controller MR_BLE IP, the Bluetooth LE link layer provides a hardware solution for the LE privacy resolution.

The process to do a LE Privacy resolution is the following:

1. The processor must provide to the AES block:
    a. the address in RAM where to find the 128-bit key array through the AESLEPRIVPOINTERREG IP_BLE APB register,
    b. the reference HASH through the AESLEPRIVHASHREG IP_BLE APB register,
    c. the random number through the AESLEPRIVPRANDREG IP_BLE APB register.
    d. the maximum key number through the AESLEPRIVCMDREG IP_BLE APB register (NBKEYS bit field).
1. Enable the AES LE privacy interrupt by setting the INTENA bit in the AESLEPRIVCMDREG IP_BLE APB register.
2. Launch the calculation by setting the Start bit in the AESLEPRIVCMDREG IP_BLE APB register (auto-cleared bit).
3. At the end of computation, the interrupt line dedicated to AES (irq_BLE_int2 also called BLE_AES) is raised and the CPU gets the reason in the INTERRUPT2REG [1] = AESLEPRIVINT bit.
4. write 1 in the INTERRUPT2REG[1] = AESLEPRIVINT bit to clear the flag.
5. The results are then available in the following registers / bit fields:
    a. the KEYFIND bit in the AESLEPRIVSTATREG IP_BLE APB register indicates if a key has been found (KEYFIND bit = 1) or not (KEYFIND) = 0 in the list,
    b. If KEYFIND = 1, the KEYFINDINDEX[7:0] bit field in the AESLEPRIVSTATREG IP_BLE APB register indicates which key of the array is the found one.

25.11.32 MSB-first feature

The MSB-first feature, not compliant with the Bluetooth standard, is added to extend to potential proprietary protocols.

The principle is:

- in TX: to swap the endianness inside transmitted bytes before whitening,
- in RX: to swap the endianness inside received bytes after dewhitening.

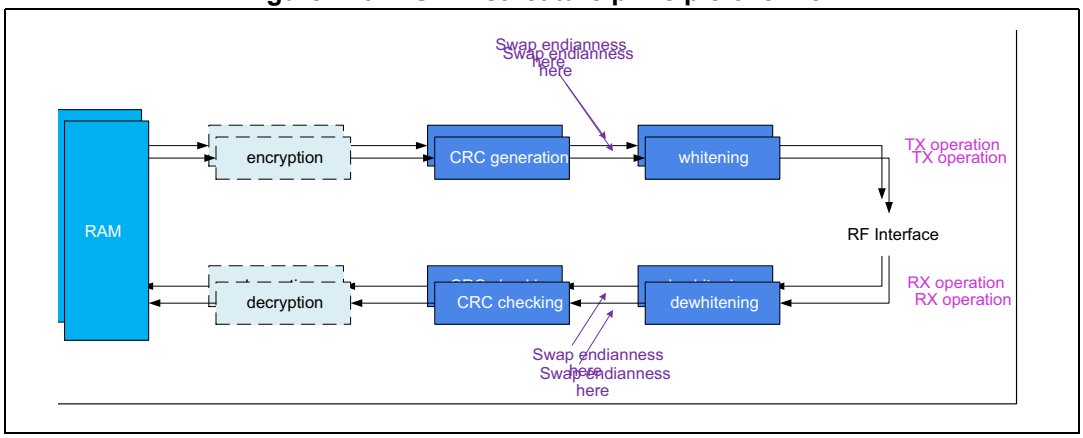
The bit endianness modification is done only on the PDU part of the packet. This means:

- the PREAMBLE is sent as usual,
- the bit endianness inversion for the Access Address must be managed at SW level by reverting the endianness of the programmed value in the StatMach.AccAddr[31:0].

However, the Radio controller MR_BLE receive block is able to manage the 2 first bytes of the PDU regardless the endianness option:

- first byte to export its content (LLID, MD, SN, NESN)
- second byte to correctly extract the length information from the HEADER1,

Figure 210. MSBFirst feature principle overview



The feature is enabled by setting the StatMach.MsbFirst bit.

This feature **must not** be enabled when at least one of the conditions below is true:

- the packet is in coded PHY format (long range),
- the CTE mode is enabled (StatMach.CTEDisable=0),
- the packet is encrypted,
- StatMach.DisableCrc = 0. Note that in this configuration, either Viterbi must be disabled or packet length=0 cannot be used.

There is no HW mechanism to ensure previous forbidden configurations are not active at the same time as the MSBFirst feature. So if the SW does not respect the rule, the integrity of the transfer is not guaranteed.

25.12 Bluetooth LE IP registers

The **BLUE_BLOCKBaseAddress** keyword used for all the register base address information corresponds to the IP_BLE registers base address decided by the SoC when integrating the IP.

*Note: **BLUE_BLOCKBaseAddress** is 0x6000_0000 in STM32WB09xE product.*

Table 145. BLE IP register list

Offset	Register Name	Register Title	Reset
0x0004	INTERRUPT1REG	INTERRUPT1REG register	0x0000 0000
0x0008	INTERRUPT2REG	INTERRUPT2REG register	0x0000 0000
0x000C	TIMEOUTDESTREG	TIMEOUTDESTREG register	0x0000 0000
0x0010	TIMEOUTREG	TIMEOUTREG register	0x0000 0000
0x0014	TIMERCAPTUREREG	TIMERCAPTUREREG register	0x0000 0000
0x0018	CMDREG	CMDREG register	0x0000 0000
0x001C	STATUSREG	STATUSREG register	0x0000 0000
0x0020	INTERRUPT1ENABLEREG	INTERRUPT1ENABLEREG register	0x0000 0000
0x0024	INTERRUPT1LATENCYREG	INTERRUPT1LATENCYREG register	0x0000 0000
0x0028	MANAESKEY0REG	MANAESKEY0REG register	0x0000 0000
0x002C	MANAESKEY1REG	MANAESKEY1REG register	0x0000 0000
0x0030	MANAESKEY2REG	MANAESKEY2REG register	0x0000 0000
0x0034	MANAESKEY3REG	MANAESKEY3REG register	0x0000 0000
0x0038	MANAESCLEARTEXT0REG	MANAESCLEARTEXT0REG register	0x0000 0000
0x003C	MANAESCLEARTEXT1REG	MANAESCLEARTEXT1REG register	0x0000 0000
0x0040	MANAESCLEARTEXT2REG	MANAESCLEARTEXT2REG register	0x0000 0000
0x0044	MANAESCLEARTEXT3REG	MANAESCLEARTEXT3REG register	0x0000 0000
0x0048	MANAESCIPHERTEXT0REG	MANAESCIPHERTEXT0REG register	0x0000 0000
0x004C	MANAESCIPHERTEXT1REG	MANAESCIPHERTEXT1REG register	0x0000 0000
0x0050	MANAESCIPHERTEXT2REG	MANAESCIPHERTEXT2REG register	0x0000 0000
0x0054	MANAESCIPHERTEXT3REG	MANAESCIPHERTEXT3REG register	0x0000 0000
0x0058	MANAESCMDREG	MANAESCMDREG register	0x0000 0000
0x005C	MANAESSTATREG	MANAESSTATREG register	0x0000 0000
0x0060	AESLEPRIVPOINTERREG	AESLEPRIVPOINTERREG register	0x0000 0000
0x0064	AESLEPRIVHASHREG	AESLEPRIVHASHREG register	0x0000 0000
0x0068	AESLEPRIVPRANDREG	AESLEPRIVPRANDREG register	0x0000 0000
0x006C	AESLEPRIVCMDREG	AESLEPRIVCMDREG register	0x0000 0000
0x0070	AESLEPRIVSTATREG	AESLEPRIVSTATREG register	0x0000 0000
0x007C	STATUS2REG	STATUS2REG register	0x0000 0000

25.12.1 INTERRUPT1REG register (INTERRUPT1REG)

Address offset: 0x0004

Reset value: 0x0000 0000 (0xFFFF FFFF)

INTERRUPT1REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCVOK	RCVCRCERR	TIMECAPTURETRIG	RCVCMD	RCVNOMD	RCVTIMEOUT	DONE	TXOK	CONFIGERROR	ACTIVE2ERROR	TXRXSKIP	Res.	SEMATIMEOUTERROR	RCVLENGTHERROR	Res.	NOACTIVELERROR
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1		rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDATAREADYERROR	ALLTABLEREADYERROR	ENCERROR	TXERROR_4	TXERROR_3	TXERROR_2	TXERROR_1	TXERROR_0	SEQDONE	Res.	RXOVERFLOWERROR	ADDPOINTERROR	Res.	Res.	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1				

Bit 31 **RCVOK**: Receive data OK.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 30 **RCVCRCERR**: Receive data fail
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 29 **TIMECAPTURETRIG**: A time has been captured in TIMERCAPTUREREG.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 28 **RCVCMD**: Received command
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 27 **RCVNOMD**: Received low MD bit.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 26 **RCVTIMEOUT**: Receive timeout (no preamble found).
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 25 **DONE**: Receive/Transmit done.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 24 **TXOK**: Previous transmitted packet received OK by the peer device.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 23 **CONFIGERROR**: Data pointer configuration error.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 22 **ACTIVE2ERROR**: Active2 Radio state error.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 21 **TXRXSKIP**: Transmission/Reception skip.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **SEMATIMEOUTERROR**: Semaphore timeout error
When read, indicates the interrupt.
Write 1'b1 to clear.

Bit 18 **RCVLENGTHERROR**: Receive length error.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **NOACTIVELERROR**: GlobalStatMach.active bit error.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 15 **TXDATAREADYERROR**: Transmit data pack not ready error
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 14 **ALLTABLEREADYERROR**: All RAM Table not ready on time.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 13 **ENCERROR**: Encryption error on reception.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 12 **TXERROR_4**: Transmission error 4. A CTE issue occurred.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 11 **TXERROR_3**: Transmission error 3. Error while waiting for the confirmation the Radio FSM is in TX state.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 10 **TXERROR_2**: Transmission error 2. Channel index is greater than 39.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 9 **TXERROR_1**: Transmission error 1. A TX skip happened during an on-going transmission.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 8 **TXERROR_0**: Transmission error 0. Transmit block missing data error.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 7 **SEQDONE**: sequencer end of task.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **RXOVERFLOWERROR**: Receive Overflow error.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 4 **ADDPOINTERROR**: Address Pointer Error.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bits 3:0 Reserved, must be kept at reset value.

Note: All bits with the corresponding enable mask at '0' are seen at '0' in this register whatever the status.

The full unmasked status is visible in the STATUSREG register. Refer to Section 25.12.7: STATUSREG register (STATUSREG) for an exhaustive flag description.

25.12.2 INTERRUPT2REG register (INTERRUPT2REG)

Address offset: 0x0008

Reset value: 0x0000 0000 (0xFFFF FFFF)

INTERRUPT2REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AESLEPRIVINT	AESMANENCINT
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **AESLEPRIVINT**: AES LE privacy engine.
This interrupt is enabled through MANAESCMDREG register.
When read, indicates the interrupt status.
Write 1'b1 to clear.

Bit 0 **AESMANENCINT**: AES manual encryption.
This interrupt is enabled through AESLEPRIVCMDREG register.
When read, indicates the interrupt status.
Write 1'b1 to clear.

25.12.3 TIMEOUTDESTREG register (TIMEOUTDESTREG)

Address offset: 0x000C

Reset value: 0x0000 0000 (0xFFFF FFFF)

TIMEOUTDESTREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DESTINATION[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **DESTINATION[1:0]**: Timeout timer Destination

- 00 or 01: all disabled
- 10: Timer1 enable
- 11: Timer2 enable (but Timer2 really starts counting at the end of a Rx/Tx sequence)

Note:

- Enabling one of the two timers automatically disables the second one.

25.12.4 TIMEOUTREG register (TIMEOUTREG)

Address offset: 0x0010

Reset value: 0x0000 0000 (0xFFFF FFFF)

TIMEOUTREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMEOUT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TIMEOUT[31:0]**: Timer1 or Timer2 Timeout value (depending on Destination register)

Time units:

- In microseconds for timer 2.
- in periods of a 512 Khz clock for timer 1.

25.12.5 TIMERCAPTUREREG register (TIMERCAPTUREREG)

Address offset: 0x0014

Reset value: 0x0000 0000 (0xFFFF FFFF)

TIMERCAPTUREREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMERCAPTURE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMERCAPTURE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TIMERCAPTURE[31:0]**: Interpolated absolute time capture register

See. TxRxPack.TrigRcv/TrigDone, GlobalStatMach.TimeCapture/TimeCaptureSel for detailed specification.

This register is cleared on the beginning of a new Bluetooth LE sequence (sequencer trigger event).

Time unit is in 16 x slow clock so typically 512 kHz period cycle.

25.12.6 CMDREG register (CMDREG)

Address offset: 0x0018

Reset value: 0x0000 0000 (0xFFFF FFFF)

CMDREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLEARSEMAREQ	Res.	Res.	TXRXSKIP
												w			w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CLEARSEMAREQ**: Semaphore Clear command.

Setting this bit releases the token for the IP_BLE. Software option in parallel of the hardware management by the BLE sequencer through TxRxPack.KeepSemaReq bit.

This bit is auto cleared by the HW.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **TXRXSKIP**: Transmission/Reception skip command.

This bit is auto cleared by the HW.

25.12.7 STATUSREG register (STATUSREG)

Address offset: 0x001C

Reset value: 0x0000 0000 (0xFFFF FFFF)

STATUSREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCVOK	RCVCRCERR	TIMECAPTURETRIG	RCVCMD	RCVNOMD	RCVTIMEOUT	DONE	TXOK	CONFIGERROR	ACTIVE2ERROR	TXRXSKIP	Res.	SEMATIMEOUTERROR	RCVLENGTHERROR	Res.	NOACTIVELERROR
r	r	r	r	r	r	r	r	r	r	r		r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDATAREADYERROR	ALLTABLEREADYERROR	ENCERROR	TXERROR_4	TXERROR_3	TXERROR_2	TXERROR_1	TXERROR_0	SEQDONE	PREVTRANSMIT	RXOVERFLOWERROR	ADDPOINTERROR	NOTSUPPORTED_FUNCTION	Res.	Res.	AESONFLYBUSY
r	r	r	r	r	r	r	r	r	r	r	r	r			r

Bit 31 **RCVOK**: Receive data OK status

Bit 30 **RCVCRCERR**: Receive data fail (CRC error or invalid CI field) status.
Note: This error is raised only if at least preamble and access address have been detected.

Bit 29 **TIMECAPTURETRIG**: indicates a time has been captured in TIMERCAPTUREREG when set.

Bit 28 **RCVCMD**: Received command status (valid only on Data Physical Channel PDU reception).
This flag is raised when LLID = 2'b11 in the received data packet header.

Bit 27 **RCVNOMD**: Received MD bit status (valid only on Data Physical Channel PDU reception)
This flag is raised when MD = 0 in the received data packet header.

Bit 26 **RCVTIMEOUT**: Receive timeout status (no access address found)

Bit 25 **DONE**: Receive/Transmit done status.
This flag is set if the sequencer reached the Transmission/Reception step.

Bit 24 **TXOK**: Previous transmitted packet received OK by the peer device status. This bit is updated at the end of a reception.
0: the previous transmitted packet was not received OK by the peer device.
1: the previous transmitted packet was received OK by the peer device.
This bit is set only if the following conditions are verified:
- this is a data packet,
- the SN/NESN mechanism is enabled (TxRxPack.SN_EN = 1),
- a preamble and a good access address have been received inside the receive window,
- the received NESN is different from the local StatMach.SN bit.

Bit 23 **CONFIGERROR**: Data pointer configuration error status

Bit 22 **ACTIVE2ERROR**: Indicates the Radio FSM was not in ACTIVE2 state when the sequencer reaches the end of 1st INIT step.

Bit 21 **TXRXSKIP**: Transmission/Reception skip status.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **SEMATIMEOUTERROR**: Semaphore timeout error status

This flag is raised when the IP_BLE token request is not granted on time by the RRM semaphore

Bit 18 **RCVLENGTHERROR**: Receive length error status

Bit 17 Reserved, must be kept at reset value.

Bit 16 **NOACTIVELERROR**: GlobalStatMach.active bit error status

This flag is raised when the sequencer reads GlobalStatMach.active = 0 at the beginning of a trigged sequence

Bit 15 **TXDATAREADYERROR**: Transmit data pack not ready status.

Indicates the data to transmit are not ready in RAM when TX on antenna is about to start.

This flag is raised if the sequencer reads TxRxPack.TxdataReady = 0 during DATA INIT step (for transmission sequence only)

Bit 14 **ALLTABLEREADYERROR**: All RAM Table not ready status

Bit 13 **ENCERROR**: Encryption error on receive status

Bit 12 **TXERROR_4**: Transmission error 4 status

Possible causes are:

- the CTETime field is not between 2 and 20 inclusive: the transmission applied the CTE anyway (informative flag)
- or in case of CTE enabled with a coded packet: the transmission occurs without CTE

Bit 11 **TXERROR_3**: Transmission error 3.

Error while waiting for the confirmation the Radio FSM is in TX state (timeout defined in GlobalStatMach.TxReadyTimeout[7:0] bit field).

Possible causes are:

- the Radio FSM encounters issue and did not go in TX state (e.g. PLL lock failure)
- the TxReadyTimeout[7:0] delay is too short

Bit 10 **TXERROR_2**: Transmission error 2 status.

This flag is raised if the requested channel number is greater than 39.

This channel index comes:

-directly from SW in RAM table when TxRxPack.IncChan = 0,

-from the channel incrementer block when TxRxPack.IncChan = 1

Note: the channel index used for the failing TX can be read in StateMach.Remap_chan at the end of the sequence.

Bit 9 **TXERROR_1**: Transmission error 1 status

This flag is raised if a TxSkip event occurs during the Transmission/Reception step of the sequencer (mainly due to a SW skip through CMDREG.TXRXSKIP bit or possibly due to a PLL lock issue during EN_PA Radio FSM state).

Bit 8 **TXERROR_0**: Transmission error 0 status. Transmit block missing data error.
This flag is raised if the transmit block did not receive bytes to transmit from RAM on time during transmission).
Note: on this error, the transmit block stops the on-going transmission but the sequencer manages it as a normal end of transmission. This TXERROR_0 flag is the only information available for the user regarding this issue.

Bit 7 **SEQDONE**: Sequencer end of task status.
This bit is set each time the sequencer ends the execution of a sequence due to a trigger event whatever the result (OK, with errors, ACTIVE bit not set, etc.).

Bit 6 **PREVTRANSMIT**: Previous event was a Transmission (1) or Reception (0) status

Bit 5 **RXOVERFLOWERROR**: AHB arbiter is full and there is no more storage capability available in RX datapath

Bit 4 **ADDPOINTERROR**: Address Pointer Error status
This flag is set when the MSB[31:24] part of some address pointers defined in the RAM tables is not equal to the MSB[31:24] part of the RAM base address of the device.

Bit 3 **NOTSUPPORTED_FUNCTION**: indicates the SW requests an unsupported feature.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **AESONFLYBUSY**: AES on the fligh encryption busy status

Note: This register is updated on each sequencer end of sequence.

This register is cleared each time the sequencer starts a new sequence (timer trig event) except for the bit tagged with (*):

CONFIGERROR: updated when the sequencer reads the StatMach

PREVTRANSMIT: updated when the sequencer reaches the transmission/reception step again

After a reception, a SN_NESN error is identified if the STATUSREG indicates the RX is done (DONE=1), not OK (RCVOK=0) but no other specific error flag is set.

When a transmission occurred well, the DONE flag (in STATUSREG and potentially INTERRUPT1REG) and the STATUSREG.PREVTRANSMIT bit are set.

25.12.8 INTERRUPT1ENABLEREG register (INTERRUPT1ENABLEREG)

Address offset: 0x0020

Reset value: 0x0000 0000 (0xFFFF FFFF)

INTERRUPT1ENABLEREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCVOK	RCVCRCERR	TIMECAPTURETRIG	RCVCMD	RCVNOMD	RCVTIMEOUT	DONE	TXOK	CONFIGERROR	ACTIVE2ERROR	TXRXSKIP	Res.	SEMATIMEOUTERROR	RCVLENGTHERROR	Res.	NOACTIVELERROR
r	r	r	r	r	r	r	r	r	r	r		r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDATAREADYERROR	ALLTABLEREADYERROR	ENCERROR	TXERROR_4	TXERROR_3	TXERROR_2	TXERROR_1	TXERROR_0	SEQDONE	Res.	RXOVERFLOWERROR	ADDPOINTERROR	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r		r	r				

Bit 31 **RCVOK**: Receive data OK enable interruption

Bit 30 **RCVCRCERR**: Receive data fail enable interruption

Bit 29 **TIMECAPTURETRIG**: TimerCaptureReg time capture enable interruption

Bit 28 **RCVCMD**: Received command enable interruption

Bit 27 **RCVNOMD**: Received MD bit embedded in the PDU data packet header was zero enable interruption

Bit 26 **RCVTIMEOUT**: Receive timeout enable interruption (no preamble found)

Bit 25 **DONE**: Receive/Transmit done interruption

Bit 24 **TXOK**: Previous transmitted packet received OK enable interruption

Bit 23 **CONFIGERROR**: Data pointer configuration error enable interruption

Bit 22 **ACTIVE2ERROR**: Active2 Radio state error enable interruption

Bit 21 **TXRXSKIP**: Transmission/Reception skip enable interruption

Bit 20 Reserved, must be kept at reset value.

Bit 19 **SEMATIMEOUTERROR**: Semaphore timeout error enable interruption

Bit 18 **RCVLENGTHERROR**: Receive length error enable interruption

Bit 17 Reserved, must be kept at reset value.

Bit 16 **NOACTIVELERROR**: active bit error enable interruption

Bit 15 **TXDATAREADYERROR**: Transmit data pack not ready enable interruption

Bit 14 **ALLTABLEREADYERROR**: All RAM Table not ready enable interruption

Bit 13 **ENCERROR**: Encryption error on receive enable interruption

Bit 12 **TXERROR_4**: Transmission error 4 enable interruption

Bit 11 **TXERROR_3**: Transmission error 3 enable interruption

Bit 10 **TXERROR_2**: Transmission error 2 enable interruption

Bit 9 **TXERROR_1**: Transmission error 1 enable interruption

Bit 8 **TXERROR_0**: Transmission error 0 enable interruption

Bit 7 **SEQDONE**: Sequencer end of task enable interruption

Bit 6 Reserved, must be kept at reset value.

Bit 5 **RXOVERFLOWERROR**: Rx Overflow Error enable interruption

Bit 4 **ADDPOINTERROR**: Address Pointer Error enable interruption

Bits 3:0 Reserved, must be kept at reset value.

*Note: This **read-only** register is a copy/summary of all the enable mask bit located in the different RAM tables treated by the sequencer.*

When '0', corresponding interrupt was masked during previous sequence.

When '1', corresponding interrupt was enabled during the previous sequence.

25.12.9 INTERRUPT1LATENCYREG register (INTERRUPT1LATENCYREG)

Address offset: 0x0024

Reset value: 0x0000 0000 (0xFFFF FFFF)

INTERRUPT1LATENCYREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERRUPT1LATENCY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTERRUPT1LATENCY[7:0]**: relative time counter started on irq_BLE_int1 (BLE_TXRX) occurrence.

Time unit: 1 µs

Clamped at 255.

Reset when all INTERRUPT1REG sources are cleared or when a new irq_BLE_int1 (BLE_TXRX) is raised.

25.12.10 MANAESKEY0REG register (MANAESKEY0REG)

Address offset: 0x0028

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESKEY0REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MANAESKEY_31_0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MANAESKEY_31_0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MANAESKEY_31_0[31:0]**: Manual mode AES key

25.12.11 MANAESKEY1REG register (MANAESKEY1REG)

Address offset: 0x002C

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESKEY1REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MANAESKEY_63_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MANAESKEY_63_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MANAESKEY_63_32[31:0]**: Manual mode AES key

25.12.12 MANAESKEY2REG register (MANAESKEY2REG)

Address offset: 0x0030

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESKEY2REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MANAESKEY_95_64[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MANAESKEY_95_64[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MANAESKEY_95_64[31:0]**: Manual mode AES key

25.12.13 MANAESKEY3REG register (MANAESKEY3REG)

Address offset: 0x0034

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESKEY3REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MANAESKEY_127_96[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MANAESKEY_127_96[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MANAESKEY_127_96[31:0]**: Manual mode AES key

25.12.14 MANAESCLEARTEXT0REG register (MANAESCLEARTEXT0REG)

Address offset: 0x0038

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCLEARTEXT0REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AES_CLEAR_31_0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AES_CLEAR_31_0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AES_CLEAR_31_0[31:0]**: Manual AES clear text

25.12.15 MANAESCLEARTEXT1REG register (MANAESCLEARTEXT1REG)

Address offset: 0x003C

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCLEARTEXT1REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AES_CLEAR_63_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AES_CLEAR_63_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AES_CLEAR_63_32[31:0]**: Manual AES clear text

25.12.16 MANAESCLEARTEXT2REG register (MANAESCLEARTEXT2REG)

Address offset: 0x0040

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCLEARTEXT2REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AES_CLEAR_95_64[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AES_CLEAR_95_64[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AES_CLEAR_95_64[31:0]**: Manual AES clear text

25.12.17 MANAESCLEARTEXT3REG register (MANAESCLEARTEXT3REG)

Address offset: 0x0044

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCLEARTEXT3REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AES_CLEAR_127_96[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AES_CLEAR_127_96[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AES_CLEAR_127_96[31:0]**: Manual AES clear text

25.12.18 MANAESCIPHERTEXT0REG register (MANAESCIPHERTEXT0REG)

Address offset: 0x0048

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCIPHERTEXT0REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AES_CIPHER_31_0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AES_CIPHER_31_0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AES_CIPHER_31_0[31:0]**: Manual AES cipher text

25.12.19 MANAESCIPHERTEXT1REG register (MANAESCIPHERTEXT1REG)

Address offset: 0x004C

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCIPHERTEXT1REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AES_CIPHER_63_32[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AES_CIPHER_63_32[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AES_CIPHER_63_32[31:0]**: Manual AES cipher text

25.12.20 MANAESCIPHERTEXT2REG register (MANAESCIPHERTEXT2REG)

Address offset: 0x0050

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCIPHERTEXT2REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AES_CIPHER_95_64[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AES_CIPHER_95_64[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AES_CIPHER_95_64[31:0]**: Manual AES cipher text

25.12.21 MANAESCIPHERTEXT3REG register (MANAESCIPHERTEXT3REG)

Address offset: 0x0054

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCIPHERTEXT3REG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AES_CIPHER_127_96[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AES_CIPHER_127_96[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AES_CIPHER_127_96[31:0]**: Manual AES Cipher Text

25.12.22 MANAESCMDREG register (MANAESCMDREG)

Address offset: 0x0058

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESCMDREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTENA	START
														rw	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **INTENA**: AES Manual encryption interrupt enable on Interrupt2Reg

Bit 0 **START**: AES Manual encryption Start command.
This bit is auto cleared by the HW.

25.12.23 MANAESSTATREG register (MANAESSTATREG)

Address offset: 0x005C

Reset value: 0x0000 0000 (0xFFFF FFFF)

MANAESSTATREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **BUSY**: AES manual encryption busy status

25.12.24 AESLEPRIVPOINTERREG register (AESLEPRIVPOINTERREG)

Address offset: 0x0060

Reset value: 0x0000 0000 (0xFFFF FFFF)

AESLEPRIVPOINTERREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POINTER[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POINTER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **POINTER[23:0]**: AES Le privacy pointer

25.12.25 AESLEPRIVHASHREG register (AESLEPRIVHASHREG)

Address offset: 0x0064

Reset value: 0x0000 0000 (0xFFFF FFFF)

AESLEPRIVHASHREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASH[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HASH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **HASH[23:0]**: AES Le privacy Reference Hash

25.12.26 AESLEPRIVPRANDREG register (AESLEPRIVPRANDREG)

Address offset: 0x0068

Reset value: 0x0000 0000 (0xFFFF FFFF)

AESLEPRIVPRANDREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRAND[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRAND[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **PRAND[23:0]**: AES Le privacy Prand

25.12.27 AESLEPRIVCMDREG register (AESLEPRIVCMDREG)

Address offset: 0x006C

Reset value: 0x0000 0000 (0xFFFF FFFF)

AESLEPRIVCMDREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	NBKEYS[7:0]								INTENA	START
						rw	rw	rw	rw	rw	rw	rw	rw	rw	w

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:2 **NBKEYS[7:0]**: AES Le privacy number of keys pointed by AesLePrivPointerReg (points to the resolution key list.)

Bit 1 **INTENA**: AES Le privacy interrupt enable on Interrupt2Reg

Bit 0 **START**: AES Le privacy Start command. Autoreset

25.12.28 AESLEPRIVSTATREG register (AESLEPRIVSTATREG)

Address offset: 0x0070

Reset value: 0x0000 0000 (0xFFFF FFFF)

AESLEPRIVSTATREG register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	KEYFNDINDEX[7:0]								KEYFND	BUSY
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:2 **KEYFNDINDEX[7:0]**: AES Le privacy index of the key found in the resolution key list.

Bit 1 **KEYFND**: AES Le privacy key finding status

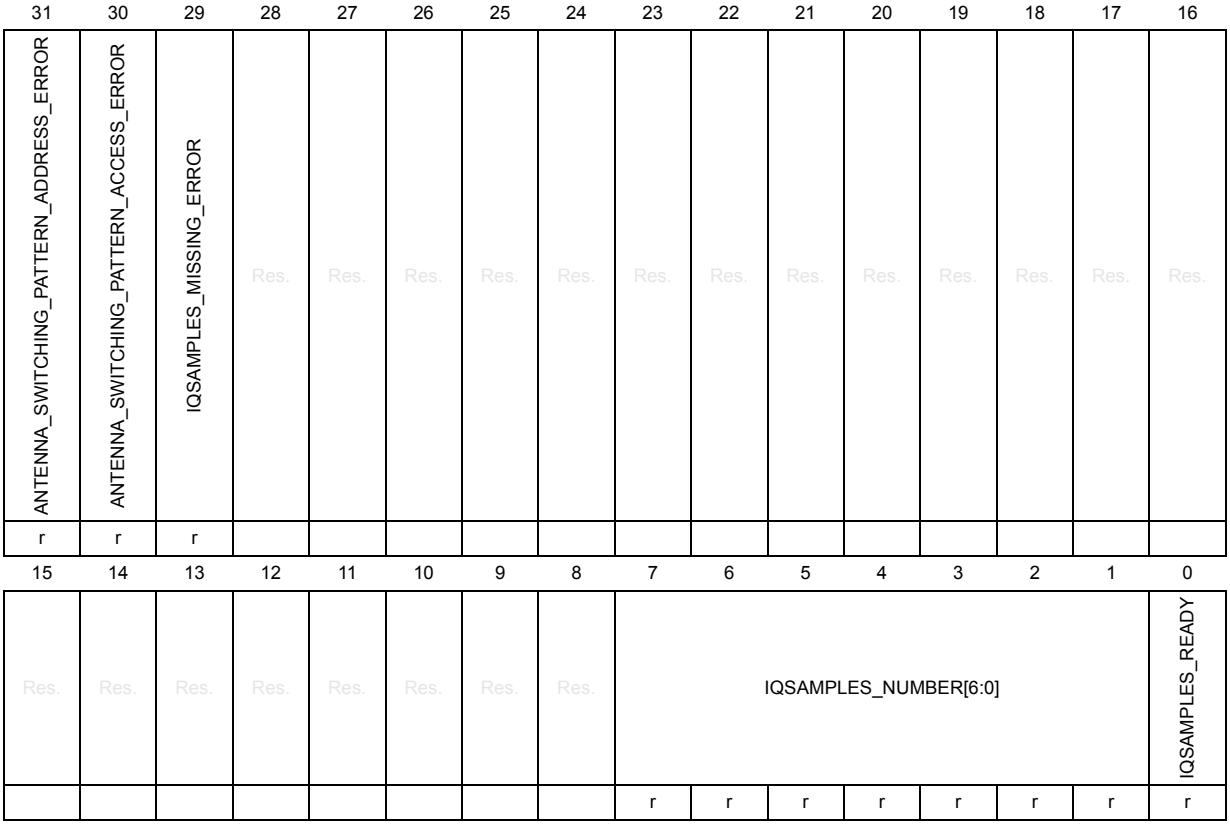
Bit 0 **BUSY**: AES Le privacy busy status

25.12.29 STATUS2REG register (STATUS2REG)

Address offset: 0x007C

Reset value: 0x0000 0000 (0xFFFF FFFF)

STATUS2REG register



Bit 31 **ANTENNA_SWITCHING_PATTERN_ADDRESS_ERROR**: AHB access error flag.

This bit is set when an internal error is raised while the IP_BLE tries to read the Antenna pattern in RAM. This indicates the value contained in the StatMach.AntennaPatternPtr is not pointing on a supported address in the SoC mapping.

Bit 30 **ANTENNA_SWITCHING_PATTERN_ACCESS_ERROR**: timing error flag related to Antenna Pattern not read on-time.

This bit is set when the HW antenna switching sub-block requests a new element of the Antenna Pattern and does not get it on-time versus antenna switching event.

Bit 29 **IQSAMPLES_MISSING_ERROR**: IQ sample internal buffer overflow error flag.

This bit is set when the internal buffer storing the IQ samples is full and new IQ samples has to be recording. The reason would be a too long latency on AHB write transfer from this internal buffer to RAM.

Bits 28:8 Reserved, must be kept at reset value.

Bits 7:1 **IQSAMPLES_NUMBER[6:0]**: indicate the number of IQ samples stored in the RAM buffer addressed by StatMach.IQSamplesPtr.

Bit 0 **IQSAMPLES_READY**: indicates if IQ samples have been received on the last reception.

0: no IQ samples reception occurred so no IQ samples stored in RAM

1: IQ samples received and stored in RAM. Exact number of stored samples is available in IQSAMPLES_NUMBER[6:0] bit field.

25.13 Wakeup block

The Wakeup block is partially located in the always-on power domain to stay supplied even in the low power modes of the device. All features not mandatory during low power modes are located in the 1.2 V switchable power domain to limit power consumption.

The Wakeup block combines in fact two features:

- wakeup / sleep requests management,
- absolute and interpolated time computation.

25.13.1 Time features management

The wakeup block computes two kinds of time: the absolute time and the interpolated time.

Absolute time

This timer is located in the always-on power domain and is based on a rollover free running counter. The absolute time is computed by a 28-bit counter clocked on the slow clock (around 32 kHz).

This absolute time is used to generate a wakeup event to the power controller block of the device when the programmed target reaches the current absolute time.

Two different targets can be programmed in parallel:

- one associated to the Bluetooth LE link layer
- one to generate a device wakeup event for CPU/application purpose

The absolute time is also used by the time interpolator block to build the 28-bit MSB non-interpolated part of the 32-bit interpolated time.

Interpolated time

The interpolated time is in the 1.2 V switchable power domain and is clocked at 16 x slow clock frequency (typically 512 kHz).

This interpolated time is a 32-bit timer built with:

- 28-bit MSB part corresponding to the non-interpolated time clocked at 32 kHz (absolute time),
- 4-bit LSB corresponding to the fractional part (interpolation at 512 kHz).

The 32-bit interpolated time is used by the Bluetooth LE link layer to get current time information and to manage the Timer1.

The 512 kHz interpolation part (4-LSB) is generated using both the 32 kHz and the system clock using a 16 MHz base whatever the system clock frequency.

Slow clock frequency statistics

A module computes the minimum, the maximum and the average value of the slow clock period length by counting the number of 16 MHz period in a slow clock period. The value is tuned on each slow clock cycle.

The calculation is done continuously from the moment the Radio controller MR_BLE IP reset is released. The result is available in MINIMUM_PERIOD_LENGTH, AVERAGE_PERIOD_LENGTH and MAXIMUM_PERIOD_LENGTH registers (see *Section 25.14: Wakeup registers*).

The average value is calculated using the previous sampled results as recursive weight: the new calculation is added to previous average and divided by the number of measurements up to 16. Then the ratio factor stays at 16.

The software can reset the minimum/maximum period value and/or the average value registers thanks to dedicated command bit in STATISTICS_RESTART register.

25.13.2 Sleep feature management

The sleep management informs the device power controller block if the Radio controller MR_BLE allows the device to go in low power mode (“sleep request”).

The Wakeup block allows the device to go in low power mode (through sleep request information) if:

- the IP_BLE sleep enable feature is active (bit SLEEP_EN = 1 in the BLE_SLEEP_REQUEST_MODE APB register),
- the IP_BLE is not busy which means:
  - the sequencer is in IDLE state (no sequence on-going),
  - Timer1 and Timer2 are not enabled,
  - no pending interrupt(s) in INTERRUPT1REG APB register (related to irq_BLE_int1 also called BLE_TXRX),

When the device is in low power, the slow clock timer is still active and may generate a wakeup request to the power controller.

25.13.3 Wakeup management

The wakeup feature is in charge to wake up the device from a low power mode and in a second time (once power and clock tree are restored), to wake up the Bluetooth LE link layer and/or the CPU.

Note: The CPU wakeup/interrupt feature offers an additional low power timer to the device.

The Wakeup block manages:

- the SoC wakeup event generation to restart the power and clock system (based on 28-bit absolute time only),
- the IP_BLE wakeup trigger event, supposed to be done once the power and accurate clock are restored (based on 32-bit interpolated time)
- a CPU wakeup interrupt, supposed to be done once the power and accurate clock are restored (based on 32-bit interpolated time)

SoC wakeup event generation

The Wakeup block offers the possibility to wake up the SoC before to wake up the Bluetooth LE link layer or the CPU to let time to power and clocks to settle.

This way, the user only has to program dynamically along its application the Bluetooth LE link layer / CPU wakeup time target, letting the HW manage the anticipated SoC wakeup.

The mechanism consists in programming a dedicated register called WAKEUP_OFFSET at the power on with a value corresponding to at least the duration of the power and clock restoration from a low power mode exit.

The HW automatically uses this information to anticipate the SoC wakeup event generation from this duration versus the absolute time wakeup target.

Information to program the SoC wakeup event time:

- a wakeup offset information must be filled in a WAKEUP_OFFSET[7:0] bit field.
- this value is in slow clock period time units (typically 32 kHz)
- So the maximum SoC power and clocks settlement time supported by the Radio controller MR_BLE IP is:
  - typically, 7.96 ms when slow clock is 32 kHz
  - 5.2 ms if slow clock is 49 kHz
  - 10.6 ms if slow clock is 24 kHz

IP_BLE wakeup management

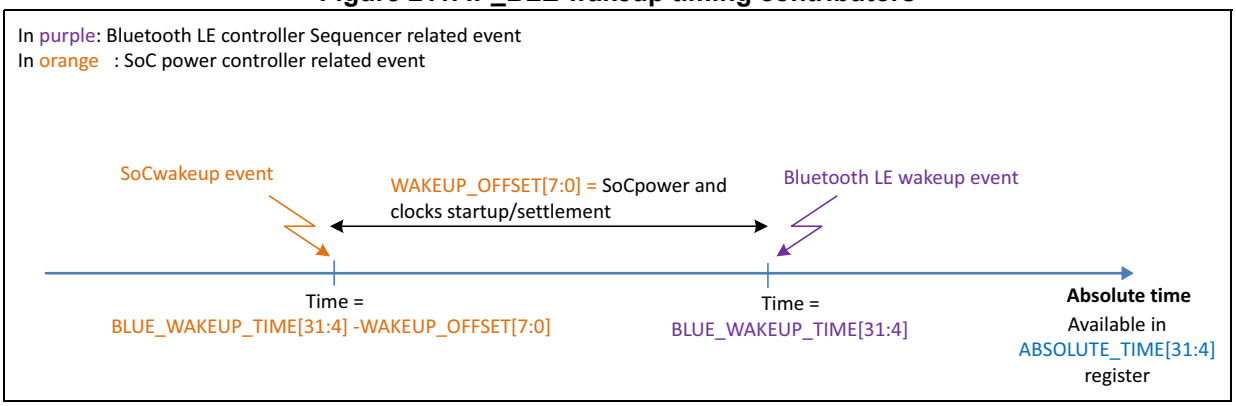
The user must program two information to activate the Bluetooth LE link layer wakeup feature:

- the BLUE_SLEEP_REQUEST_MODE[30] = BLE_WAKEUP_EN bit must be set to enable the wakeup feature,
- the Bluetooth LE link layer wakeup target time in the BLUE_WAKEUP_TIME[31:0] APB register
  - This register represents a 32-bit interpolated time
  - the 28-bit absolute time (corresponding to BLUE_WAKEUP_TIME[31:4]) is used first to generate the anticipated event to the power controller and the wakeup trigger event to the Bluetooth LE link layer sequencer,
  - the 32-bit interpolated time information (corresponding to BLUE_WAKEUP_TIME[31:0]), is used by the Bluetooth LE link layer sequencer block to manage the 1st INIT step duration

Note: The user must ensure when setting the BLE_WAKEUP_EN bit that the programmed IP_BLE wakeup time is as least WAKEUP_OFFSET[7:0] duration later.

Table 211 provides an overview of the registers involved in the wakeup management and summarize steps describes just above.

Figure 211. IP_BLE wakeup timing contributors



25.13.4 CPU wakeup management

A similar behavior is possible to generate a CPU wakeup event.

The user must program three information to activate the CPU wakeup feature:

- the CM0_SLEEP_REQUEST_MODE[30] = CM0_WAKEUP_EN bit must be set to enable the wakeup feature,
- the CPU wakeup target time in the CM0_WAKEUP_TIME[31:4] APB register
- the WAKEUP_CM0_IRQ_ENABLE[0] = WAKEUP_IT bit must be set to have an interrupt generated on CPU on event

Note: Only the 28-bit absolute time is used for the CPU wakeup feature, so granularity of wakeup target is slow clock frequency.

In this case, the wakeup process also occurs in two steps:

- At ABSOLUTE_TIME[31:4] = CM0_WAKEUP_TIME [31:4] - WAKEUP_OFFSET[7:0]
  - The Wakeup block raises a SoC wakeup request towards the Power Controller of the SoC to request voltage/clock restoring.
- At ABSOLUTE_TIME[31:4] = CM0_WAKEUP_TIME [31:4]
  - if WAKEUP_CM0_IRQ_ENABLE.WAKEUP_IT = 1, the Wakeup block sends an interrupt towards the CPU (irq_wakeup_cpu line)

This feature allows using the existing slow clock timer to generate a CPU wakeup source. This feature when activated has no impact on the Bluetooth LE transfers (no trigger event generated to the IP_BLE sequencer)

25.14 Wakeup registers

The **WAKEUP_SLEEP_BLOCKBaseAddress** keyword used for all the register base address information corresponds to the Wakeup registers base address decided by the SoC when integrating the IP.

*Note: **WAKEUP_SLEEP_BLOCKBaseAddress** is 0x6000_1800 in STM32WB09xE product.*

Table 146. Wakeup register list

Offset	Register Name	Register Title	Reset
0x0008	WAKEUP_OFFSET	WAKEUP_OFFSET register	0x0000 0000
0x0010	ABSOLUTE_TIME	ABSOLUTE_TIME register	0x0000 0000
0x0014	MINIMUM_PERIOD_LENGTH	MINIMUM_PERIOD_LENGTH register	0x0000 0000
0x0018	AVERAGE_PERIOD_LENGTH	AVERAGE_PERIOD_LENGTH register	0x0000 0000
0x001C	MAXIMUM_PERIOD_LENGTH	MAXIMUM_PERIOD_LENGTH register	0x0000 0000
0x0020	STATISTICS_RESTART	STATISTICS_RESTART register	0x0000 0000
0x0024	BLUE_WAKEUP_TIME	BLUE_WAKEUP_TIME register	0x0000 0000
0x0028	BLUE_SLEEP_REQUEST_MODE	BLUE_SLEEP_REQUEST_MODE register	0x0000 0007
0x002C	CM0_WAKEUP_TIME	CM0_WAKEUP_TIME register	0x0000 0000
0x0030	CM0_SLEEP_REQUEST_MODE	CM0_SLEEP_REQUEST_MODE register	0x8000 0007
0x0040	WAKEUP_BLE_IRQ_ENABLE	WAKEUP_BLE_IRQ_ENABLE register	0x0000 0000
0x0044	WAKEUP_BLE_IRQ_STATUS	WAKEUP_BLE_IRQ_STATUS register	0x0000 0000
0x0048	WAKEUP_CM0_IRQ_ENABLE	WAKEUP_CM0_IRQ_ENABLE register	0x0000 0000
0x004C	WAKEUP_CM0_IRQ_STATUS	WAKEUP_CM0_IRQ_STATUS register	0x0000 0000

25.14.1 WAKEUP_OFFSET register (WAKEUP_OFFSET)

Address offset: 0x0008

Reset value: 0x0000 0000 (0xFFFF FFFF)

Offset used to anticipate the SOC wakeup event

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_OFFSET[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WAKEUP_OFFSET[7:0]**: delay of anticipation of the Soc device to settle power and clock
Unit is in slow clock period time (typically 32 kHz)

25.14.2 ABSOLUTE_TIME register (ABSOLUTE_TIME)

Address offset: 0x0010

Reset value: 0x0000 0000 (0xFFFF FFFF)

Absolute Time register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABSOLUTE_TIME[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABSOLUTE_TIME[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ABSOLUTE_TIME[31:0]**: absolute time
This value is expressed in number of (slow_clock *16) frequency period cycle (typically 512 kHz).
Note: ABSOLUTE_TIME[31:4] is clocked on the slow clock (typically 32 kHz), ABSOLUTE_TIME[3:0] is the interpolation at slow clock * 16 frequency (typically 512 kHz).

25.14.3 MINIMUM_PERIOD_LENGTH register (MINIMUM_PERIOD_LENGTH)

Address offset: 0x0014

Reset value: 0x0000 0000 (0xFFFF FFFF)

Minimum period length register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	LENGTH[9:0]										Res.	Res.	Res.	Res.
		r	r	r	r	r	r	r	r	r	r				

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:4 **LENGTH[9:0]**: minimum period length computed by Time Interpolator

Bits 3:0 Reserved, must be kept at reset value.

25.14.4 AVERAGE_PERIOD_LENGTH register (AVERAGE_PERIOD_LENGTH)

Address offset: 0x0018

Reset value: 0x0000 0000 (0xFFFF FFFF)

average period length computed by Time Interpolator

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AVERAGE_COUNT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	LENGTH_INT[9:0]										LENGTH_FRACT[3:0]			
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **AVERAGE_COUNT[7:0]**: Number of slow clock cycles.
This value indicates the number of slow clock periods considered to calculate the average.
This bit field is updated every slow clock period.
This bit field is clamped at 0xFF so reading 0xFF means at least 128 slow clock periods ar already been used to calculate the average.

Bits 23:14 Reserved, must be kept at reset value.

Bits 13:4 **LENGTH_INT[9:0]**: average period length computed by Time Interpolator.
This value indicates the number of 16 MHz clock cycles contained in 1 slow clock period.
This bit field is updated every 16 slow clock periods.

Bits 3:0 **LENGTH_FRACT[3:0]**: additional information/precision on slow clock frequency.
Reading AVERAGE_PERIOD_LENGTH[13:0] indicates the number of 16 MHz clock cycles contained in 16 slow clock periods.
This bit field is updated every 16 slow clock periods.

25.14.5 MAXIMUM_PERIOD_LENGTH register (MAXIMUM_PERIOD_LENGTH)

Address offset: 0x001C

Reset value: 0x0000 0000 (0xFFFF FFFF)

Maximum period length register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	LENGTH[9:0]										Res.	Res.	Res.	Res.
		r	r	r	r	r	r	r	r	r	r				

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:4 **LENGTH[9:0]**: maximum period length computed by Time Interpolator

Bits 3:0 Reserved, must be kept at reset value.

25.14.6 STATISTICS_RESTART register (STATISTICS_RESTART)

Address offset: 0x0020

Reset value: 0x0000 0000 (0xFFFF FFFF)

Statistics restart register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_AVR	CLR_MIN_MAX
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CLR_AVR**: Write '1' to clear the AVERAGE_PERIOD_LENGTH register value.
This action clears both the average length value and the average counter.
Note: this bit is auto cleared by the HW.

Bit 0 **CLR_MIN_MAX**: Write '1' to clear the minimum and maximum registers.
Note: this bit is auto cleared by the HW.

25.14.7 BLUE_WAKEUP_TIME register (BLUE_WAKEUP_TIME)

Address offset: 0x0024

Reset value: 0x0000 0000 (0xFFFF FFFF)

Absolute IP_BLE wakeup time register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WAKEUP_TIME[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEUP_TIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **WAKEUP_TIME[31:0]**: programmed wakeup time for the IP_BLE.
Unit is in (16 x slow clock) period so typically 512 kHz when slow clock is 32 kHz.

25.14.8 BLUE_SLEEP_REQUEST_MODE register (BLUE_SLEEP_REQUEST_MODE)

Address offset: 0x0028

Reset value: 0x0000 0007 (0xFFFF FFFF)

IP_BLE sleep request mode register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FORCE_SLEEPING	BLE_WAKEUP_EN	SLEEP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **FORCE_SLEEPING**: IP_BLE sleeping control.
0: the IP_BLE sleeping is managed dynamically according to the IP_BLE activity and status
1: the IP_BLE is always considered as sleeping which means it always allows the SoC to go in low power mode.

Bit 30 **BLE_WAKEUP_EN**: IP_BLE wakeup enable.
0: disable the IP_BLE wakeup
1: enable the IP_BLE wakeup.
This bit is auto cleared by hardware when a wakeup event occurs (IP_BLE wakeup time matches with current time).

Bit 29 **SLEEP_EN**: IP_BLE sleeping mode enable.
0: disable IP_BLE sleeping mode = no low power mode request when the Bluetooth LE link layer indicates it is not busy.
1: enable IP_BLE sleeping mode = low power mode request when the Bluetooth LE link layer indicates it is not busy anymore.
Note: The sequencer is not busy if no sequence is on-going and if no Timer1 nor Timer2 counter is enabled (to trig the next sequence).

Bits 28:0 Reserved, must be kept at reset value.

25.14.9 CM0_WAKEUP_TIME register (CM0_WAKEUP_TIME)

Address offset: 0x002C

Reset value: 0x0000 0000 (0xFFFF FFFF)

Absolute CPU wakeup time register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WAKEUP_TIME[27:12]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEUP_TIME[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:4 **WAKEUP_TIME[27:0]**: programmed wakeup time for CPU.
Unit is in slow clock period (This is why this register bits 3:0 are always read as zero as no 512 kHz granularity on this time wakeup).

Bits 3:0 Reserved, must be kept at reset value.

25.14.10 CM0_SLEEP_REQUEST_MODE register (CM0_SLEEP_REQUEST_MODE)

Address offset: 0x0030

Reset value: 0x8000 0007 (0xFFFF FFFF)

CPU Sleep request mode register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FORCE_SLEEPING	CPU_WAKEUP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **FORCE_SLEEPING**: CPU sleeping control.

0: the CPU sleeping information is also monitored by the MR_BLE wakeup block to decide if it allows or not the low power mode at SoC level

1: the CPU is always considered as sleeping by the wakeup block.

Note: this bit must always be kept high to let the CPU WFI instruction managing alone the low power mode allowance for CPU side.

Bit 30 **CPU_WAKEUP_EN**: CPU wakeup enable.

0: disable the CPU wakeup request

1: enable the CPU wakeup request.

This bit is auto cleared by hardware when a wakeup event occurs (CM0_WAKEUP_TIME value matches with current time).

Bits 29:0 Reserved, must be kept at reset value.

25.14.11 WAKEUP_BLE_IRQ_ENABLE register (WAKEUP_BLE_IRQ_ENABLE)

Address offset: 0x0040

Reset value: 0x0000 0000 (0xFFFF FFFF)

Wakeup IP_BLE interrupt enable register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_IT
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **WAKEUP_IT**: IP_BLE wakeup interrupt enable.
0: disable the IP_BLE wakeup interrupt towards CPU.
1: enable IP_BLE wakeup interrupt towards the CPU.

25.14.12 WAKEUP_BLE_IRQ_STATUS register (WAKEUP_BLE_IRQ_STATUS)

Address offset: 0x0044

Reset value: 0x0000 0000 (0xFFFF FFFF)

Wakeup IP_BLE interrupt status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_IT
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **WAKEUP_IT**: On read, returns the IP_BLE wakeup interrupt status.
Write '1' to clear this IRQ status bit.

25.14.13 WAKEUP_CM0_IRQ_ENABLE register (WAKEUP_CM0_IRQ_ENABLE)

Address offset: 0x0048

Reset value: 0x0000 0000 (0xFFFF FFFF)

Wakeup CPU interrupt enable register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_IT
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **WAKEUP_IT**: CPU wakeup interrupt enable.
0: disable the CPU wakeup interrupt towards CPU.
1: enable CPU wakeup interrupt towards the CPU.

25.14.14 WAKEUP_CM0_IRQ_STATUS register (WAKEUP_CM0_IRQ_STATUS)

Address offset: 0x004C

Reset value: 0x0000 0000 (0xFFFF FFFF)

Wakeup CPU interrupt status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_IT
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **WAKEUP_IT**: On read, returns the CPU wakeup interrupt status.
Write '1' to clear this IRQ status bit.

Table 147. Radio IP register map and reset values

Offset	Register	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
RRM registers																																	
0x0000 to 0x000C Reserved																																	
0x0010	UDRA_CTRL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RELOAD_RDCFGPTR
	Reset value	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
0x0014	UDRA_IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD_END	CMD_START	RADIO_CFG_PTR_RELOADED
	Reset value	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
0x0018	UDRA_IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD_END	CMD_STARD	RADIO_CFG_PTR_RELOADED
	Reset value	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
0x001C	UDRA_RADIO_CFG_PTR	RADIO_CONFIG_ADDRESS[31:0																															
	Reset value	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
0x0020	SEMA_IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK	LOCK
	Reset value	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
0x0024	SEMA_IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK	LOCK
	Reset value	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
0x0028	BLE_IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_CMD_END	PORT_CMD_START	Res.	PORT_RELEASE	PORT_GRANT	Res.
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x002C	BLE_IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD_END	CMD_START	Res.	PORT_RELEASE	PORT_GRANT
	Reset value	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
0x0030 to 0x005C Reserved																																	
0x0060	VP_CPU_CMD_BUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMMAND_REQ	COMMAND[2:0]		
	Reset value	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
0x0064	VP_CPU_SEMA_BUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAKE_REQ	TAKE_PRIO[2:0]		
	Reset value	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
0x0068	VP_CPU_IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_CMD_END	PORT_CMD_START	Res.	PORT_RELEASE	PORT_GRANT
	Reset value	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
0x006C	VP_CPU_IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD_END	CMD_START	PORT_PREEMPT	PORT_RELEASE	PORT_GRANT
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
Radio registers																																	
0x0000	AA0_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA_7_0[7:0]							
	Reset value	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	1	0	1	1	0
0x0004	AA1_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA_15_8[7:0]							
	Reset value	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	0	1	1	1	1	1	0
0x0008	AA2_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA_23_16[7:0]							
	Reset value	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	0	0	0	1	0	0	1
0x000C	AA3_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AA_31_24[7:0]							
	Reset value	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	0	0	0	1	1	1	0
0x0010	DEM_MOD_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHANNEL_NUM[6:0]							Res.
	Reset value	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	1	0	0	1	1	0
0x0014	RADIO_FSM_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PA_POWER[4:0]					EN_CALIB_SYNTH	EN_CALIB_CBP	Res.
	Reset value	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	1	0	0
0x0018	PHYCTRL_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXTXPHY[2:0]		
	Reset value	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
0x001C to 0x0044 Reserved																																	
0x0048	AFC1_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFC_DELAY_BEFORE[3:0]				AFC_DELAY_AFTER[3:0]			
	Reset value	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	0	0	0	1	0	0
0x004C to 0x0050 Reserved																																	
0x0054	CR0_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR_GAIN_BEFORE[3:0]				CR_GAIN_AFTER[3:0]			
	Reset value	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	0	0	0	1	0	0
0x0058 to 0x0064 Reserved																																	

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x0068	CR0_LR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR_LR_GAIN_BEFORE[3:0]				CR_LR_GAIN_AFTER[3:0]			
	Reset value	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	1	1	0
0x006C	VIT_CONF_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPARE[5:0]						Res.	VIT_EN
	Reset value	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
0x0070 to 0x0080 Reserved																																	
0x0084	LR_PD_THR_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LR_PD_THR[7:0]							
	Reset value	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	0	1	0	1	0	1
0x0088	LR_RSSI_THR_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LR_RSSI_THR[7:0]							
	Reset value	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	1	0	0	1	1
0x008C	LR_RSSI_THR_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LR_AAC_THR[7:0]							
	Reset value	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	1	1	1	0	0	0
0x0090 to 0x00A4 Reserved																																	
0x00A8	SYNTHCAL0_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNTH_IF_FREQ_CAL[1:0]		Res.	Res.	SYNTHCAL_DEBUG_BUS_SEL[3:0]			
	Reset value	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
0x00AC to 0x00EC Reserved																																	
0x00F0	DTB5_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_SELECTED_0	PORT_SELECTED_EN	INITIALIZE	RX_ACTIVE	TX_ACTIVE	RXTX_START_SEL
	Reset value	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
0x00F4 to 0x0044 Reserved																																	

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x0148	RXADC_ANA_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXADC_DELAYTRIM_Q_TST_SEL	RXADC_DELAYTRIM_I_TST_SEL	RFD_RXADC_DELAYTRIM_Q[2:0]			RFD_RXADC_DELAYTRIM_I[2:0]		
	Reset value	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	1	1	0	1	1
0x014C to 0x0150 Reserved																																	
0x0154	LDO_ANA_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_RF_REG_BYPASS
	Reset value	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
0x0158 to 0x0170 Reserved																																	
0x0174	LDO_ANA_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFD_CBIAS_IPTAT_TRIM[3:0]				RFD_CBIAS_IBIAS_TRIM[3:0]			
	Reset value	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	0	0	0	1	0	0	0
0x0178	CBIAS1_ANA_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBIAS0_TRIM_TST_SE	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	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
0x017C Reserved																																	
0x0180	SYNTHCAL0_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALAMP_OUT_6_0[6:0]							Res.
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x0184	SYNTHCAL1_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALAMP_OUT_10_7[3:0]			
	Reset value	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	1
0x0188	SYNTHCAL2_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCO_CALFREQ_OUT[6:0]						
	Reset value	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	1	0	0
0x018C	SYNTHCAL3_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNTHCAL_DEBUG_BUS[7:0]							
	Reset value	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
0x0190	SYNTHCAL4_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOD_REF_DAC_WORD_OUT[5:0]					
	Reset value	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	1	1	0	0	0
0x0194	SYNTHCAL5_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBP_CALIB_WORD[3:0]			
	Reset value	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	1	1	1
0x0198	FSM_STATUS_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNTH_CAL_ERROR	Res.	Res.	STATUS[4:0]				
	Reset value	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
0x019C to 0x01A0 Reserved																																	
0x01A4	RSSI0_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSI_MEAS_OUT_7_0[7:0]							
	Reset value	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	1	0	0	0
0x01A8	RSSI1_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSI_MEAS_OUT_15_8[7:0]							
	Reset value	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	1	0	0	0
0x01AC	AGC_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_ATT_OUT[3:0]			
	Reset value	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
0x01B0	DEMOD_DIG_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_END	PD_FOUND	AAC_FOUND	CI_FIELD[1:0]	
	Reset value	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
0x01B4 to 0x01B8 Reserved																																	

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x01BC	AGC2_ANA_TST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_ANTENNAE_USR_TRIM[2:0]			AGC2_ANA_TST_SEL
	Reset value	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
0x01C0	AGC0_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_ENABLE	AGC_THR_HIGH[5:0]					
	Reset value	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	0	0	1	0	1	0
0x01C4	AGC1_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AGC_LOCK_SYNC	AGC_AUTOLOCK	AGC_THR_LOW_6[5:0]					
	Reset value	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	0	0	0	0	1	0	0
0x01C8 to 0x01E0 Reserved																																	
0x01E8	AGC10_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_0[1:0]		ATT_LNA_0		ATT_IF_0[2:0]	
	Reset value	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
0x01EC	AGC11_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_1[1:0]		ATT_LNA_1		ATT_IF_1[2:0]	
	Reset value	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
0x01F0	AGC12_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_2[1:0]		ATT_LNA_2		ATT_IF_2[2:0]	
	Reset value	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	1	0	0	0	0	0
0x01F4	AGC13_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_3[1:0]		ATT_LNA_3		ATT_IF_3[2:0]	
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x01F8	AGC14_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_4[1:0]		ATT_LNA_4	ATT_IF_4[2:0]		
	Reset value	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	1	1	1	0	0	0
0x01FC	AGC15_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_5[1:0]		ATT_LNA_5	ATT_IF_5[2:0]		
	Reset value	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	1	1	1	0	0	1
0x0200	AGC16_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_6[1:0]		ATT_LNA_6	ATT_IF_6[2:0]		
	Reset value	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	1	1	1	0	1	0
0x0204	AGC17_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_7[1:0]		ATT_LNA_7	ATT_IF_7[2:0]		
	Reset value	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	1	1	1	0	1	1
0x0208	AGC18_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_8[1:0]		ATT_LNA_8	ATT_IF_8[2:0]		
	Reset value	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	1	1	1	1	0	0
0x020C	AGC19_DIG_ENG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATT_ANT_9[1:0]		ATT_LNA_9	ATT_IF_9[2:0]		
	Reset value	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	1	1	1	1	0	1
0x0210 to 0x0220 Reserved																																	
0x0224	RXADC_HW_TRIM_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_RXADC_DELAYTRIM_Q[2:0]			HW_RXADC_DELAYTRIM_I[2:0]		
	Reset value	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	1	1	0	1	1

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x0228	CBIAS0_HW_TRIM_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_CBIAS_IPTAT_TRIM[3:0]				HW_CBIAS_IBIAS_TRIM[3:0]			
	Reset value	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	0	0	0	1	0	0	0
0x022C Reserved																																	
0x0230	AGC_HW_TRIM_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_AGC_ANTENNAE_TRIM[2:0]			Res.
	Reset value	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	1	1	0
0x0234 to 0x0238 Reserved																																	
0x0240	ANTSW0_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TIME_TO_SAMPLE[6:0]							Res.
	Reset value	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	1	1	1	0	0
0x0244	ANTSW1_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX_TIME_TO_SWITCH[5:0]						
	Reset value	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	1	0	1	1
0x0248	ANTSW2_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TX_TIME_TO_SWITCH[6:0]							
	Reset value	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	0	1	0	0	1	
0x024C	ANTSW3_DIG_USR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TX_TIME_TO_SWITCH_2M[6:0]							
	Reset value	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	1	0	0	1	1	

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
Radio controller registers																																	
0x0000	RADIO_CONTROL_ID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRODUCT[3:0]				VERSION[3:0]				REVISION[3:0]				Res.	Res.	Res.	Res.
	Reset value	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	0	0	0	0	0	0	0
0x0004	CLK32COUNT_REG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_COUNT[8:0]								
	Reset value	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	1	0	1	1	1
0x0008	CLK32PERIOD_REG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_PERIOD[18:0]																		
	Reset value	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
0x000C	CLK32FREQUENCY_REG	Res.	Res.	Res.	Res.	Res.	SLOW_FREQUENCY[26:0]																										
	Reset value	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
0x0010	RADIO_CONTROL_IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADIO_FSM_IRQ[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_CLK_IRQ
	Reset value	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
0x0014	RADIO_CONTROL_IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADIO_FSM_IRQ_MASK[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLOW_CLK_IRQ_MASK
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
GlobalStatMach registers																																	
0x0000	WORD0	RADIOCONFIGPTR[31:0]																															
	Reset value	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
0x0004	WORD1	TIMER2INITDELAYNOCAL[7:0]								TIMER12INITDELAYCAL[7:0]								WAKEUPINITDELAY[7:0]								ACTIVE	CURSTMACHNUM[6:0]							
	Reset value	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
0x0008	WORD2	RECEIVENOCALDELAYCHK[7:0]								RECEIVECALDELAYCHK[7:0]								TRANSMITNOCALDELAYCHK[7:0]								TRANSMITCALDELAYCHK[7:0]							
	Reset value	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
0x000C	WORD3	TIMECAPTURE	TIMECAPTURESEL	TXDELAYEND[5:0]						TXDELAYSTART[7:0]								TXDATAREADYCHECK[7:0]								CONFIGENDDURATION[7:0]							
	Reset value	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
0x0010	WORD4	INTCONFIGERROR	INTACTIVE2ERR	INTTXRXSKIP	INTSEQDONE	Res.	INTSEMATIMEOUTERROR	INTRCVLENGTHERROR	Res.	INTNOACTIVELERROR	INTTXDATAREADYERROR	INTALLTABLEREADYERROR	INTADDPOINTERROR	Res.	Res.	Res.	Res.	INTCONFIGERROR	INTACTIVE2ERR	INTTXRXSKIP	INTSEQDONE	Res.	INTSEMATIMEOUTERROR	INTRCVLENGTHERROR	Res.	INTNOACTIVELERROR	INTTXDATAREADYERROR	INTALLTABLEREADYERROR	INTADDPOINTERROR	Res.	Res.	Res.	Res.
	Reset value	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
0x0014	WORD5	INTCONFIGERROR	INTACTIVE2ERR	INTTXRXSKIP	INTSEQDONE	Res.	INTSEMATIMEOUTERROR	INTRCVLENGTHERROR	Res.	INTNOACTIVELERROR	INTTXDATAREADYERROR	INTALLTABLEREADYERROR	INTADDPOINTERROR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHKFLAGAUTOCLEARENA	Res.	AUTOTXRXSKIPEN
	Reset value	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
0x0018	WORD6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEFAULTANTENNAID[6:0]						
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
StatMach registers																																	
0x0000	WORD0	Res.	RXPHY[2:0]		CTEDISABLE	TXPHY[2:0]			RCVENC	TXENC	ENCRYPTON	BUFFER_FULL	Res.	Res.	PHYSCHANPDUTYPE[1:0]		NESN	SN	REMAP_CHAN[5:0]						TXMODE	RADIOCOMLISTENA	UCHAN[5:0]						
	Reset value	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
0x0004	WORD1	TXPOINT[31:0]																															
	Reset value	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	1	1
0x0008	WORD2	RCVPOINT[31:0]																															
	Reset value	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
0x000C	WORD3	RCVPOINTPREV[31:0]																															
	Reset value	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
0x0010	WORD4’	TXPOINTNEXT[31:0]																															
	Reset value	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
0x0014	WORD5	PCNTTX_31_0[31:0]																															
	Reset value	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
0x0018	WORD6	PCNTRCV_23_0[23:0]																								PCNTTX_39_32[7:0]							
	Reset value	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
0x001C	WORD7	RXDEBUGCRC	INTRXOVERFLOWERROR	INTENCERROR	INTTXERROR[4:0]					RXMICDBG	MSBFIRST	DISABLECRC	ENAPREAMBLEREP	PREAMBLEREP[3:0]				PCNTRCV_39_24[15:0]																	
	Reset value	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
0x0020	WORD8	RXDEBUGCRC	INTRXOVERFLOWERROR	INTENCERROR	INTTXERROR[4:0]					RXMICDBG	MSBFIRST	DISABLECRC	ENAPREAMBLEREP	PREAMBLEREP[3:0]				PCNTRCV_39_24[15:0]																	
	Reset value	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
0x0024	WORD9	ACCADDR[31:0]																															
	Reset value	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
0x0028	WORDA	MAXRECEIVEDLENGTH[7:0]								CRCINIT[23:16]								CRCINIT[15:0]															
	Reset value	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
0x002C	WORDB	USEDCHANNELFLAGS_15_0[15:0]																Res.	Res.	HOPINCR[5:0]						TXHP	Res.	Res.	PAPOWER[4:0]				
	Reset value	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
0x0030	WORDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USEDCHANNELFLAGS_36_16[21:16]						USEDCHANNELFLAGS_36_16[15:0]															
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x0034	WORDD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVENTCOUNTER[15:0]															
	Reset value	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
0x0038	WORDE	ENCRYPTIV_31_0[31:0]																															
	Reset value	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
0x003C	WORDF	ENCRYPTIV_63_32[31:0]																															
	Reset value	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
0x0040	WORD10	ENCRYPTK_31_0[31:0																															
	Reset value	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
0x0044	WORD11	ENCRYPTK_63_32[31:0]																															
	Reset value	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
0x0048	WORD12	ENCRYPTK_95_64[31:0]																															
	Reset value	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
0x004C	WORD13	ENCRYPTK_127_96[31:0]																															
	Reset value	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
0x0050	WORD14	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANTENNAPATTERNLENGTH[7:0]								Res.	MAXIMUMIQSAMPLESNUMBER[6:0]							Res.	CTETIME[4:0]					CTESLOTWIDTH	AOD_NAOA
	Reset value	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
0x0054	WORD15	IQSAMPLESPTR[31:0]																															
	Reset value	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
0x0058	WORD16	ANTENNAPATTERNPTR[31:0]																															
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
TxRxPack registers																																	
0x0000	WORD0	NEXTPTR[31:0]																															
	Reset value	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
0x0004	WORD1	Res.	Res.	SUBEVENTCHANALGO2	DISABLEWHITENING	Res.	TXDATAREADY	ALLTABLEREADY	NEXTTXMODE	INCCHAN	SN_EN	ADVERTISE	CRCINITSEL	CTEANDSAMPLINGENABLE	KEEPSEMAREQ	CHANALGO2SEL	CALREQ	Res.	Res.	SUBEVENTCHANALGO2	DISABLEWHITENING	Res.	TXDATAREADY	ALLTABLEREADY	NEXTTXMODE	INCCHAN	SN_EN	ADVERTISE	CRCINITSEL	CTEANDSAMPLINGENABLE	KEEPSEMAREQ	CHANALGO2SEL	CALREQ
	Reset value	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
0x0008	WORD2	DATAPTR[31:0]																															
	Reset value	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
0x000C	WORD3	INTRCVOK	INTRCVCRCERR	INTTIMECAPTURE	INTRCVCMD	INTRCVNOMD	INTRCVTIMEOUT	INTDONE	INTTXOK	TRIGDONE	TRIGRCV	Res.	TIMER2EN	TIMER2[19:16]				TIMER2[15:0]																	
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
BLE IP registers																																	
0x0004	INTERRUPT1REG	RCVOK	RCVCRCERR	TIMECAPTURETRIG	RCVCMD	RCVNOMD	RCVTIMEOUT	DONE	TXOK	CONFIGERROR	ACTIVE2ERROR	TXRXSKIP	Res.	SEMATIMEOUTERROR	RCVLENGTHERROR	Res.	NOACTIVELERROR	TXDATAREADYERROR	ALLTABLEREADYERROR	ENCERROR	TXERROR_4	TXERROR_3	TXERROR_2	TXERROR_1	TXERROR_0	SEQDONE	Res.	RXOVERFLOWERROR	ADDPOINTERROR	Res.	Res.	Res.	Res.
	Reset value	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
0x0008	INTERRUPT2REG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AESLEPRIVINT	AESMANENCINT
	Reset value	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
0x000C	TIMEOUTDESTREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DESTINATION[1:0]	
	Reset value	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
0x0010	TIMEOUTREG	TIMEOUT[31:0]																															
	Reset value	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
0x0014	TIMERCAPTURERE G	TIMERCAPTURE[31:0]																															
	Reset value	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
0x0018	CMDREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLEARSEMAREQ	Res.	Res.	TXRXSKIP
	Reset value	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
0x001C	STATUSREG	RCVOK	RCVCRCERR	TIMECAPTURETRIG	RCVCMD	RCVNOMD	RCVTIMEOUT	DONE	TXOK	CONFIGERROR	ACTIVE2ERROR	TXRXSKIP	Res.	SEMATIMEOUTERROR	RCVLENGTHERROR	Res.	NOACTIVELERROR	TXDATAREADYERROR	ALLTABLEREADYERROR	ENCERROR	TXERROR_4	TXERROR_3	TXERROR_2	TXERROR_1	TXERROR_0	SEQDONE	PREVTRANSMIT	RXOVERFLOWERROR	ADDPOINTERROR	NOTSUPPORTED_FUNCTION	Res.	Res.	AESONFLYBUSY
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x0020	INTERRUPT1ENABLEREG	RCVOK	RCVCRCERR	TIMECAPTURETRIG	RCVCMD	RCVNOMD	RCVTIMEOUT	DONE	TXOK	CONFIGERROR	ACTIVE2ERROR	TXRXSKIP	Res.	SEMATIMEOUTERROR	RCVLENGTHERROR	Res.	NOACTIVELERROR	TXDATAREADYERROR	ALLTABLEREADYERROR	ENCERROR	TXERROR_4	TXERROR_3	TXERROR_2	TXERROR_1	TXERROR_0	SEQDONE	Res.	RXOVERFLOWERROR	ADDPOINTERROR	Res.	Res.	Res.	Res.
	Reset value	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
0x0024	INTERRUPT1LATENCYREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERRUPT1LATENCY[7:0]							
	Reset value	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
0x0028	MANAESKEY0REG	MANAESKEY_31_0[31:0]																															
	Reset value	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
0x002C	MANAESKEY1REG	MANAESKEY_63_32[31:0]																															
	Reset value	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
0x0030	MANAESKEY2REG	MANAESKEY_95_64[31:0]																															
	Reset value	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
0x0034	MANAESKEY3REG	MANAESKEY_127_96[31:16]																															
	Reset value	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
0x0038	MANAESCLEARTEXT0REG	AES_CLEAR_31_0[31:0]																															
	Reset value	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
0x003C	MANAESKEY1REG	AES_CLEAR_63_32[32:0]																															
	Reset value	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
0x0040	MANAESCLEARTEXT2REG	AES_CLEAR_95_64[31:0]																															
	Reset value	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
0x0044	MANAESCLEARTEXT3REG	AES_CLEAR_127_96[31:0]																															
	Reset value	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
0x0048	MANAESCIPHERTEXT0REG	AES_CIPHER_31_0[31:0]																															
	Reset value	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
0x004C	MANAESCIPHERTEXT1REG	AES_CIPHER_63_32[31:0]																															
	Reset value	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
0x0050	MANAESCIPHERTEXT2REG	AES_CIPHER_95_64[31:0]																															
	Reset value	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
0x0054	MANAESCIPHERTEXT3REG	AES_CIPHER_127_96[31:0]																															
	Reset value	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
0x0058	MANAESCMDREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTENA	START
	Reset value	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
0x005C	MANAESSTATREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUS
	Reset value	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
0x0060	AESLEPRIVPOINTERREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POINTER[23:0]																							
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x0064	AESLEPRIVHASHREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASH[23:0]																							
	Reset value	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
0x0068	AESLEPRIVPRANDREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRAND[23:0]																							
	Reset value	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
0x006C	AESLEPRIVCMDREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBKEYS[7:0]								INTENA	START
	Reset value	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
0x0070	AESLEPRIVSTATREG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYFNDINDEX[7:0]								KEYFND	BUSY
	Reset value	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
0x0074 to 0x0078 Reserved																																	
0x007C	STATUS2REG	ANTENNA_SWITCHING_PATTERN_ADDRESS_ERROR	ANTENNA_SWITCHING_PATTERN_ACCESSERROR	IQSAMPLES_MISSING_ERROR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IQSAMPLES_NUMBER[6:0]							IQSAMPLES_READY
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
Wakeup block registers																																	
0x0000	WAKEUP_RESERVED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	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
0x0008	WAKEUP_OFFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_OFFSET[7:0]							
	Reset value	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
0x0010	ABSOLUTE_TIME	ABSOLUTE_TIME[31:0]																															
	Reset value	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
0x0014	MINIMUM_PERIOD_LENGTH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENGTH[9:0]										Res.	Res.	Res.	Res.
	Reset value	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
0x0018	AVERAGE_PERIOD_LENGTH	AVERAGE_COUNT[7:0]							Res.	Res.	LENGTH_INT[9:0]										LENGTH_FRACT[3:0]			Res.	Res.	Res.	LENGTH_INT[9:0]						
	Reset value	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
0x001C	MAXIMUM_PERIOD_LENGTH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENGTH[9:0]										Res.	Res.	Res.	Res.
	Reset value	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
0x0020	STATISTICS_RESTART	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_AVR	CLR_MIN_MAX
	Reset value	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
0x0024	BLUE_WAKEUP_TIME	WAKEUP_TIME[31:16]																															
	Reset value	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
0x0028	BLUE_SLEEP_REQUEST_MODE	FORCE_SLEEPING	BLE_WAKEUP_EN	SLEEP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	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	1	0	0
0x002C	CM0_WAKEUP_TIME	WAKEUP_TIME[27:12]																												Res.	Res.	Res.	Res.
	Reset value	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
0x0030	CM0_SLEEP_REQUEST_MODE	FORCE_SLEEPING	CPU_WAKEUP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	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	1	0	0
0x0040	WAKEUP_BLE_IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_IT
	Reset value	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
0x0044	WAKEUP_BLE_IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_IT
	Reset value	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

Table 147. Radio IP register map and reset values (continued)

Offset	Register	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
0x0048	WAKEUP_CM0_IRQ_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_IT
	Reset value	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
0x004C	WAKEUP_CM0_IRQ_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAKEUP_IT
	Reset value	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

26 Debug support (DBG)

26.1 SWD debug features

The STM32WB09xE device JTAG ID[31:0] is the following:

– 0000 0010000000110010 00000100000 1

– (0x0203 2041)

The Cortex-M0+ subsystem of the STM32WB09xE embeds 4 breakpoints and 2 watchpoints.

The STM32WB09xE device embeds:

- the Arm serial wire debug port which enables Serial Wire debug (2-wire) to be connected to the CPU (default after power-on reset).

Note: When the device enters Deepstop mode, the SWD debug port is not powered. As a consequence, debug access is disabled and the chip cannot be accessed through SWD channel.
One possible recovery option is to activate the internal embedded UART bootloader through the PA10 pin (just force PA10 high during hardware reset).

27 Device electronic signature (DESIG)

The device electronic signature is stored in the System memory area of the flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match the characteristics of the microcontroller.

27.1 DESIG registers

27.1.1 DESIG ADC trimming max diff (DESIG_ADCMAXDIFF)

Address offset: 0x000

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset maximum differential (ADC_VINP - ADC_VINM at 1.2 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain maximum differential (ADC_VINP - ADC_VINM at 1.2 V).

27.1.2 DESIG ADC trimming max negative (DESIG_ADCMAXNEG)

Address offset: 0x004

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset maximum negative (ADC_VINM at 1.2 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain maximum negative (ADC_VINM at 1.2 V).

27.1.3 DESIG ADC trimming max positive (DESIG_ADCMAXPOS)

Address offset: 0x008

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset maximum positive (ADC_VINP at 1.2 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain maximum positive (ADC_VINP at 1.2 V).

27.1.4 DESIG ADC trimming mean diff (DESIG_ADCMEANDIFF)

Address offset: 0x00C

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset mean differential (ADC_VINP - ADC_VINM at 2.4 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain mean differential (ADC_VINP - ADC_VINM at 2.4 V).

27.1.5 DESIG ADC trimming mean negative (DESIG_ADCMEANNEG)

Address offset: 0x010

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset mean negative (ADC_VINM at 2.4 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain mean negative (ADC_VINM at 2.4 V).

27.1.6 DESIG ADC trimming max positive (DESIG_ADCMEANPOS)

Address offset: 0x014

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset mean positive (ADC_VINP at 2.4 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain mean positive (ADC_VINP at 2.4 V).

27.1.7 DESIG ADC trimming min diff (DESIG_ADCMINDIFF)

Address offset: 0x018

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset minimum differential (ADC_VINP - ADC_VINM at 3.6 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain minimum differential (ADC_VINP - ADC_VINM at 3.6 V).

27.1.8 DESIG ADC trimming min negative (DESIG_ADCMINNEG)

Address offset: 0x01C

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset minimum negative (ADC_VINM at 3.6 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain minimum negative (ADC_VINM at 3.6 V).

27.1.9 DESIG ADC trimming min positive (DESIG_ADCMINPOS)

Address offset: 0x020

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:16]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:12]				GAIN[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OFFSET[19:12]**: ADC trimming offset minimum positive (ADC_VINP at 3.6 V).

Bits 11:0 **GAIN[11:0]**: ADC trimming gain minimum positive (ADC_VINP at 3.6 V).

27.1.10 DESIG reference temperature register (DESIG_TSREFR)

Address offset: 0x05C

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS_REF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS_REF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TS_REF[31:0]**: reference temperature for ADC at 30°C.

27.1.11 DESIG temperature calibration register (DESIG_TSCAL1R)

Address offset: 0x060

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS_CAL[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS_CAL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TS_CAL[31:0]**: temperature measurement calibration for ADC at 30°C.

27.1.12 DESIG package data register (DESIG_PKGR)

Address offset: 0x0EC

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PKG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PKG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PKG[31:0]**: Package type:
0x5F325F32: VFQFPN32
0xAC36AC36: WLCSP36

27.1.13 DESIG 64-bit unique device identifier register 1 (DESIG_UID64R1)

Address offset: 0x0F0

Reset value: 0xXXXX XXXX (X is factory-programmed)

Note: The unique device identifier is a sequential number, different for each individual device.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]**: unique serial number (first 4 bytes)

27.1.14 DESIG 64-bit unique device identifier register 2 (DESIG_UID64R2)

Address offset: 0x0F4

Reset value: 0xXXXX XXXX (X is factory-programmed)

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

Bits 31:0 **UID[63:32]**: unique serial number (last 4 bytes)

27.1.15 DESIG register map

The DESIG base address location is 0x1000 1E00.

Table 148. DESIG register map and reset values

Offset	Register name Reset value	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
0x000	DESIG_ADCMAXDIFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x004	DESIG_ADCMAXNEG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x008	DESIG_ADCMAXPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x00C	DESIG_ADCMEANDIFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x010	DESIG_ADCMEANNEG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x014	DESIG_ADCMEANPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x018	DESIG_ADCMINDIFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x01C	DESIG_ADCMINNEG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x020	DESIG_ADCMINPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[19:12]								GAIN[11:0]											
	Reset value													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x024 - 0x5B	Reserved	Res.																															
0x05C	DESIG_TSREFR	TS_REF[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x060	DESIG_TSCAL1R	TS_CAL[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x064 - 0xE8	Reserved	Res.																															
0x0EC	DESIG_PKGR	PKG [31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0F0	DESIG_UID64R1	UID[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0F4	DESIG_UID64R2	UID[63:32]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

28 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

29 Revision history

Table 149. Document revision history

Date	Revision	Changes
21-Jun-2024	1	Initial release.
25-Nov-2024	2	Updated: – *Section 2.1: System architecture* – *Section 6.6.2: Internal clock source calibration register (RCC_ICSCR)* – *Section 7.3.2: I/O pin alternate function multiplexer and mapping* (added EVENTOUT description) – *Table 18: GPIO register map and reset values* – *Table 12.2.1: Temperature sensor subsystem* – Bookmark levels of *Section 12.2.4: Steady state input impedance Section 12.2.5: Input signal sampling transient response* (promoted from unnumbered headings). – *Figure 23: ADC sampling time Tsw and sampling period Ts* – *Section 12.2.7: Down sampler (DS)* (replaced “decimation filter” with “down sampler”. – *Section 12.6.8: ADC gain and offset correction 1 register (COMP_1)* through *Section 12.6.11: ADC gain and offset correction 4 register (COMP_4)* (cross-reference to DESIG registers section) – *Section 11.5.1: DMAMUX request line multiplexer Channel x Configuration Register DMAMUX register x (DMAMUX_CxCR)* – *Section 12.5.1: ADC mode overviewSection 14.4: TRNG functional description* – *Section 14.7.18: TRNG Interrupt Control Register (TRNG_IRQ_CR)* – *Section 14.7.19: Interrupt Status Register (TRNG_IRQ_SR)* Added: – *Section 12.2.2: Battery sensor* – *Section 12.2.6: Calibration points*.
04-Apr-2025	3	Updated: – References to Bluetooth standard. – Corrected references to Deepstop mode – *Figure 8: Power regulators and SMPS configuration in Shutdown mode* – *Section 6.6: RCC registers* – *Section 9: Embedded flash memory (FLASH)* – *Table 24: System memory protection* – *Section 9.4.5: Enabling protection example* – *Section 21: Inter-integrated circuit (I2C) interface*