PRELIMINARY DATA

SuperH^{TM} (SH) 32-Bit RISC Series

SH-4, ST40 System Architecture, Volume 2: Bus Interfaces

Last updated 9 May 2003 1:38 pm

Issued by the MCDT Documentation Group on behalf of STMicroelectronics

Contents

Preface

This document is part of the SuperH Documentation suite detailed below. Comments on this or other manuals in the SuperH Documentation Suite should be made by contacting your local STMicroelectronics Limited Sales Office or distributor.

Document identification and control

Each book carries a unique identifier in the form:

ADCS nnnnnnnx

Where, nnnnnnn is the document number and x is the revision.

Whenever making comments on a document the complete identification ADCS nnnnnnnx should be quoted.

ST40 documentation suite

The ST40 documentation suite comprises the following volumes:

ST40 Micro Toolset User's Guide

ADCS 7379953. This manual provides an introduction to the ST40 Micro Toolset and instructions for getting a simple OS21 application run on an STMicroelectronics' MediaRef platform. It also describes how to boot OS21 applications from ROM and how to port applications which use STMicroelectronics' STLite/OS20 operating systems to OS21.

OS21 User’s Manual

ADCS 7358306. This manual describes the generic use of OS21 across supported platforms. It describes all the core features of OS21and their use and details the OS21 function definitions.It also explains how OS21 differs to STLite/OS20, the API targeted at ST20.

OS21 for ST40 User Manual

ADCS 7358673. This manual describes the use of OS21 on ST40 platforms. It describes how specific ST40 facilities are exploited by the OS21 API. It also describes the OS21 board support packages for ST40 platforms.

32-Bit RISC Series, SH-4 CPU Core Architecture

ADCS 7182230. This manual describes the architecture and instruction set of the SH4-1xx (previously known a ST40-C200) core as used by STMicroelectronics.

32-Bit RISC Series, SH-4, ST40 System Architecture

This manual describes the ST40 family system architecture. It is split into four volumes:

ST40 System Architecture - Volume 1 System - *ADCS 7153464*.

ST40 System Architecture - Volume 2 Bus Interfaces - *ADCS 7171720*.

ST40 System Architecture - Volume 3 Video Devices - *ADCS 7225754*.

ST40 System Architecture - Volume 4 I/O Devices - *ADCS 7225754*.

Conventions used in this guide

General notation

The notation in this document uses the following conventions:

- **`Sample code`**, **`keyboard input`** and **`file names`**,
- *Variables* and ***`code variables`***,
- `Equations and math`,
- **Screens**, **windows** and **dialog boxes**,
- **Instructions**.

Hardware notation

The following conventions are used for hardware notation:

- REGISTER NAMES and FIELD NAMES,
- PIN NAMES and SIGNAL NAMES.

Software notation

Syntax definitions are presented in a modified Backus-Naur Form (BNF). Briefly:

1 Terminal strings of the language, that is those not built up by rules of the language, are printed in teletype font. For example, **`void`**.

2 Nonterminal strings of the language, that is those built up by rules of the language, are printed in italic teletype font. For example, ***`name`***.

3 If a nonterminal string of the language starts with a nonitalicized part, it is equivalent to the same nonterminal string without that nonitalicized part. For example, **`vspace-`*****`name`***.

4 Each phrase definition is built up using a double colon and an equals sign to separate the two sides.

5 Alternatives are separated by vertical bars (‘**|**’).

6 Optional sequences are enclosed in square brackets (‘**[**’ and ‘**]**’).

7 Items which may be repeated appear in braces (‘**{**’ and ‘**}**’).

1 Local memory interface (LMI)

1.1 Introduction

The local memory interface (LMI) provides the interface between the ST40, the SuperHyway and the external main memory subsystem.It comprises a SuperHyway port and an SDRAM controller. The following descriptions highlight the key features of the main memory array, SDRAM controller and SuperHyway Port.

1.1.1 Main memory organization

- The array is organized as rows.
- Each row consists of one or more discrete devices or DIMM (single or double sided) modules arranged in sockets on a PCB.

1.1.2 SDRAM controller features

SDRAM controller comprises:

- programmable external bus width: 16-, 32- and 64-bit,
- dual or quad bank SDRAM, specifically the PC-SDRAM standard, or Double Data Rate (DDR) SDRAM (types cannot be mixed in the same system),
- main memory size: from 2 Mbytes to 2 Gbytes,
- memory modules supported: two rows of discrete SDRAM, single and double density DIMMs,
- SDRAM technology: 16-, 64-, 128- and 256-Mbit,
- SDRAM speed: 66, 100 and 133 MHz.

1.1.3 SuperHyway port

The SuperHyway port includes:

- two 4-deep-in-order queue for requests and responses, respectively: supports pipelining of up to eight outstanding transactions on the SuperHyway,
- one control block (16-Mbyte space), containing the LMI module's VCR and SDRAM control registers
- 127 data blocks (16 Mbytes each) with access routed to the external memory.

1.2 SuperHyway interface

The functionality of the SuperHyway interface is described in the following subsections.

1.2.1 SuperHyway port

The SuperHyway port is divided into one control block and [n] number of data blocks. Each block is 16 Mbytes in size. The address range of the LMI is defined by VCR.BOT_MB and VCR.TOP_MB. Data blocks populate from VCR.BOT_MB up to VCR.TOP_MB (exclusive). The control block is assigned to 16 Mbytes, starting from VCR.TOP_MB (inclusive). The control block contains the LMI module's VCR and SDRAM control registers.

	LMI control block	**LMI data blocks**
Port name	LMI_CB	LMI_DB
Lowest address	0x0F00 0000	0x0800 0000
Highest address	0x0FFF FFFF	0x0EFF FFFF

Table 1: LMI address space

Note: *In the ST40 implementation,* VCR.BOT_MB = *0x08 and* VCR.TOP_MB = *0x0F. Therefore data blocks populate the address range from 0x0800 0000 to 0x0EFF FFFF. This is summarized in Table 1*

When the LMI is active, memory accesses from the SuperHyway to the data blocks cause accesses to be made on the external memory bus. Each external memory access consists of a number of phases, each one representing a specific action performed on the external memory bus. The behavior of the external pins of the LMI can be programmed to allow the LMI to drive the external memory bus in an appropriate way for different DDR SDRAM and SDRAM parts.

1.2.2 Control block

12 registers (64 bits each) are populated in the control block. Except VCR, each register contains no more than 32 defined bits. The following table summarizes all registers implemented in ST40's LMI module.

Note: VCR.TOP_MB *is 8-bit. Base Address = 0x(*VCR.TOP_MB*)000000.*

Register name	Description	Type	Address offset from 0x0F00 0000	Access size
LMI.VCR	Version control register, see *Table 11 on page 30*	RW	0x00 0000	32 or 64
LMI.MIM	Memory interface mode, see *Table 14 on page 36*	RW	0x00 0008	32 or 64
LMI.SCR	SDRAM control, see *Table 15 on page 40*	RW	0x00 0010	32 or 64
LMI.STR	SDRAM timing, see *Table 16 on page 43*	RW	0x00 0018	32 or 64
LMI.PBS	Pin buffer strength, see datasheet		0x00 0020	
LMI.COC	Clock and pad control, see datasheet	RW	0x00 0028	32 or 64
LMI.SDRA[0:1]	SDRAM row attribute, see *Table 17 on page 47*	RW	0x00 0030 to 0x00 0038	32 or 64

Table 2: Control block registers

Register name	Description	Type	Address offset from 0x0F00 0000	Access size
LMI.CIC	Clock and pad status, see datasheet	RO	0x00 0040	32 or 64
Reserved			0x00 0042 to 0x00 0080	
LMI.SDMR[0:1]	SDRAM mode register, see *Section 1.4.6: SDRAM row mode registers (LMI.SDMR[0:1]) on page 49*	WO	0x8x xxxx 0x9x xxxx	32 or 64
Undefined			Remaining	

Table 2: Control block registers

Note: If the LMI is active, the transactions to the control block are processed only when there are no outstanding data block transactions. While the LMI is processing control block transactions, the SDRAMs are in idle state. After processing control block transactions, the LMI's DRAM controller then continues with its normal behavior which reflects the state of the control registers. The LMI ensures that the change from the original behavior to the subsequent behavior is achieved instantaneously at a boundary between SDRAM commands during the processing of that transaction.

1.2.3 Reaction to packets

The LMI does not initiate request packets to the SuperHyway. The LMI processes the following packets received from SuperHyway.

For accessing the control block:

- load 1/2/4/8-byte,[1]
- store 1/2/4/8-byte.[1]

For accessing the data block:

- load 1/2/4/8-byte,[1]
- load 16-byte (burst),[2]
- load 32-byte (burst),
- store1/2/4/8-byte,[1]
- store 16-byte (burst),[3]
- store 32-byte (burst),
- SWAP 4/8-byte,[3]
- Read-Modify-Write 4/8-byte.[4]

When accessing the control block, the LMI observes the rules below.

1. LMI treats Load 1/2/4-byte and Store 1/2/4-byte as Load 8-byte and Store 8-byte. LMI performs read/write bytes according to eight byte-enable (or byte-mask) signals from the SuperHyway interface, regardless of Load1/2/4/8-byte and Store 1/2/4/8-byte. It is the responsibility of the SuperHyway initiator to assert correct byte-enable bits to ensure consistency with the intention of Load 1/2/4/8-byte and Store 1/2/4/8-byte.
2. LMI expects and performs wrap-around within a 32-byte range for Store32-byte and Load32-byte, respectively. In case of Load and Store 16-byte, LMI expects they are all 16-byte aligned. LMI raises the error flag in VCR when it detects a non-aligned load/store16 SuperHyway packet
3. LMI treats both SWAP 4-byte and SWAP 8-byte as SWAP 8-byte. It is the responsibility of the SuperHyway initiator to assert correct byte-enable bits to ensure the consistency with the intention of SWAP 4-byte and SWAP 8-byte.

- Reads from the reserved control registers return 0. Writes to the reserved control registers are ignored.
- Reads from an undefined control register return an undefined value. Writes to undefined control registers are ignored.
- Accesses to the LMI's undefined control registers result in an error bit being set to indicate an access to a bad address. The full behavior of the transactions serviced by the LMI's control block is shown in *Table 3*.

Packet received	Condition	Effect
Load 1/2/4/8-byte	Request is a subword (less than 8-byte) or whole word load from an undefined control register	VCR.PERR.BAD_ADDR set VCR.PERR.ERR_SNT set Access ignored, error response packet sent
	Request is a subword or whole word load from a reserved control register	An ordinary response sent Return data all zeros
	Request is a subword or whole-word load from an defined control register	An ordinary response sent Return data determined by control register accessed

Table 3: Packets directed to LMI's control block

4. For supporting Read-Modify-Write, LMI receives a LOCK signal from the SuperHyway. LMI treats Read-Modify-Write as Load 8-byte and Store 8-byte when LOCK = 1 and LOCK = 0, respectively. LMI performs read or write bytes according to eight byte-enable (or byte-mask) signals from the SuperHyway interface. LMI treats both Read-Modify-Write 4-byte and Read-Modify-Write 8-byte as Read-Modify-Write 8-byte. It is the responsibility of the SuperHyway initiator to assert correct byte-enable bits to ensure the consistency with the intention of SWAP 4-byte and SWAP 8-byte.

Packet received	Condition	Effect
Store 1/2/4/8-byte	Request is a subword or whole-word store to an undefined control register	VCR.PERR.BAD_ADDR set VCR.PERR.ERR_SNT set Access ignored, error response packet sent
	Request is a subword or whole-word store to a reserved control register	An ordinary response sent Access ignored
	Request is a subword or whole-word store to a defined control register	An ordinary response sent. Written data determined by control register accessed
All other packets	Request is to an undefined control register	VCR.PERR.BAD_OPC set VCR.PERR.BAD_ADDR set. VCR.PERR.ERR_SNT set Access ignored, error response packet sent
	Request is to a reserved or defined control register	VCR.PERR.BAD_OPC set VCR.PERR.ERR_SNT set. Access ignored, error response packet sent

Table 3: Packets directed to LMI's control block

When the data block is addressed, the LMI observes the rules below.

- The LMI does not service any packet directed to the data block when the SDRAM controller is disabled (MIM.DCE = 0).
- An out-of-range address is defined as a location beyond the address defined in the SDRAM row attribute register, see *Section 1.4.5: SDRAM row attribute registers (LMI.SDRA[0:1]) on page 47* for details. Reads from an out-of-range address return an undefined value. Writes to an out-of-range address are ignored. In either cases, the LMI responds with an error packet and set error flags in the VCR.The full behavior of the transactions serviced by the LMI's data block is shown in *Table 4*.

Packet received	Condition	Effect
Load 1-, 2-, 4- and 8-byte, Load 32-byte, Store 1-, 2-, 4-, and 8-byte, Store 32-byte, SWAP, Read-Modify-Write	DRAM controller disabled	VCR.MERR.ERR_SNT set VCR.MERR.DRAM_INACTIVE set Access ignored, error response packet sent
	DRAM controller enabled Address within the range	An external memory access made An ordinary response sent Effect of external memory accesses and any returned data depends on the external implementation
	DRAM controller enabled Address out of the range	VCR.MERR.ERR_SNT set VCR.MERR.BAD_ADDR set Access ignored, error response packet sent

Table 4: Behavior of the transactions serviced by the LMI's data block

Packet received	Condition	Effect
Load16-byte, Store 16-byte	DRAM controller disabled	VCR.MERR.ERR_SNT set VCR.MERR.DRAM_INACTIVE set Access ignored, error response packet sent
	DRAM controller enabled Address within the range	External memory access made Ordinary response sent Effect of external memory accesses and any returned data depends on the external implementation
	DRAM controller enabled Address out of the range or not aligned at 16-byte boundary	VCR.MERR.ERR_SNT set VCR.MERR.BAD_ADDR set Access ignored, error response packet sent
All other packets	DRAM controller disabled	VCR.MERR.ERR_SNT set VCR.MERR.DRAM_INACTIVE set Access ignored, error response packet sent
	DRAM controller enabled	VCR.MERR.BAD_OPC set VCR.MERR.ERR_SNT set Access ignored, error response packet sent

Table 4: Behavior of the transactions serviced by the LMI's data block

1.2.4 Pipelining request queue

The SuperHyway Port maintains two 4-deep-in-order queues, one for request and the other one for response. Each entry in these queues has a 32-byte data buffer and 32 byte-enable bits to accommodate the data to and from main memory. The request queue's entry is transferred to the response queue once it is serviced by the SDRAM controller. The response queue's entry is retired once it is output to the SuperHyway. Accesses to any given address are observed to occur in the order which they are received by the LMI. Any LMI requester will see responses in the same order as the requests. Byte-gathering for the subsequent write requests is not supported in the LMI. It is assumed that each bus initiator module on the SuperHyway performs byte-gathering itself.

1.2.5 Coherency

The memory of the LMI is coherent as viewed from the SuperHyway. All requests are processed sequentially on the LMI in the order of the receipt of those requests by LMI. However, the system can decouple the generation of response packets from the actual external memory bus accesses. For store transactions, the corresponding store response packets may not be returned to the initiator on the SuperHyway until the write operations are actually completed on the DRAM interface. Since the local system is the sole owner of the external main memory and all the requests to the same address are processed in order (as they are received from the SuperHyway interface) on the SDRAM interface, memory coherency is achieved.

A swap packet comes with store data. When processing a swap request, the LMI initiates a read transaction on the DRAM interface. Once the data is received by the SDRAM controller, a write command is issued.

1.2.6 Standby mode

Entering standby mode

The power management module asserts a STBY_REQ signal to the LMI module. From this point on, the LMI can not take new requests from the SuperHyway. The LMI continues to service all the transaction requests in the queue. Upon concluding the last request, the LMI issues a power-down command to the SDRAM, and asserts STBY_ACK to the power management module. In return, the power management module stops providing clock to the LMI. Consequently, the LMI's MCLKO pin is stopped at high.

Leaving standby mode

The power management module de-asserts STBY_REQ and the LMI responds with de-assertion of STBY_ACK. The power management module then restores the clock supply before it de-asserts STBY_REQ and MCLKO starts toggling accordingly.

1.3 SDRAM interface

The LMI's SDRAM controller can be configured to support PC-SDRAM and DDR SDRAMs. The functionality of the SDRAM interface is described in the following subsections.

1.3.1 Main memory configuration

The main memory is organized in rows. The data bus width can be programmed to 16-bit, 32-bit or 64-bit by writing MIM's BW field, see *Section 1.4.2: Memory interface mode register (LMI.MIM) on page 36*. The population on each row ranges from 2 Mbytes to 2 Gbytes. The different rows may have different size or technology of SDRAM population, but must have the same data bus width, burst length and share the same timing parameters defined in the STR register. SDRAM devices on the same row must be the same kind (for example, 4Mx16, 2-bank). Either PC SDRAM or DDR SDRAM can be supported but they cannot be mixed in main memory.

Note: The term "row" is used in two places, that is, SDRAM device's internal "row" address and main memory subsystem's "row" array. In this chapter, "row" indicates subsystem's row, while "(internal) row" means SDRAM's row address.

The upper boundary address of each row is defined in the SDRA.UBA field (SDRAM row attribute register's upper bound address). The request address [31:21] is compared to SDRA.UBA [31:21] to determine which NOT_CSA (chip selection) signal is to be asserted. A NOT_CSA signal is applied to all SDRAM devices on the same row.

Memory locations in these two rows must be contiguous in physical address space. SDRA1.UBA must be larger or equal to SDRA0.UBA. If the system consists of only one row (or DIMM), then it needs to be placed in the area corresponding to CS0 and SDRA0.UBA = SDRA1.UBA must be programmed. CS0 is asserted if the SuperHyway request access to LMI's data block and the request address [31:21] is less than SDRA0.UBA (exclusive).

SDRA0.UBA has the priority over SDRA1.UBA if they are equal. If the physical address is less than SDRA0.UBA, CS0 is asserted. If it is not less than SDRA0.UBA but is less than SDRA1.UBA, CS1 is asserted. Other cases are errors and are recorded in the VCR's error flag.

The following figure depicts a 64-bit wide, 96 Mbyte main memory subsystem. It is assumed VCR.BOT_MB = 0x08.

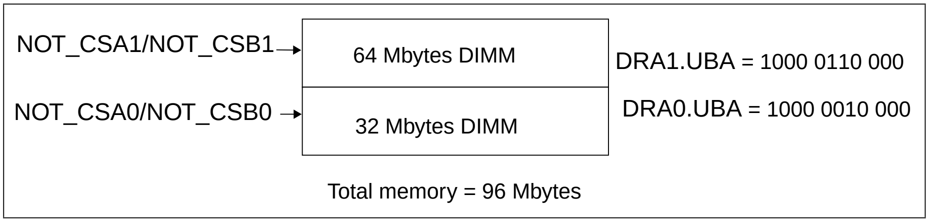


Figure 1: Main memory configuration example

Memory configuration can be little endian or big endian. The LMI is independent of endianness when the external bus width is 64 bit. When the external bus width is 32-bit, memory interface register MIM.ENDIAN bit (read only) indicates the endianness of the system.

1.3.2 SDRAM interface pins

The external pins are described in *Table 5*.

Name	I/O	Size	Description
MCLKO	Output	1	SDRAM clock out 66, 100 or 133 MHz clock output
NOT_MCLKO	Output	1	MCLKO and NOT_MCLKO are differential clock outputs to DDR SDRAM.
CKE[1:0]	Output	2	Clock enable Activates the clock signal when high and deactivates when low By deactivating the clock, CKE low initiates the power-down mode, self-refresh mode or suspend mode.
NOT_CSA[1:0] NOT_CSB[1:0]	Output	2 2	Chip select Perform the function of selecting the particular SDRAM components during the active state[A]
NOT_WEA NOT_WEB	Output	1 1	Write enable signal WE asserted during writes to SDRAM
MA[14:0]	Output	15	Row and column address
BA[1:0]	Output	2	Bank address
MD[63:0]	I/O	64	Memory data
DQS[7:0]	I/O	8	Input/output data strobe Used in DDR SDRAM only These pins provide the read and write data strobe signal to/from the receiver circuit of DRAM controller. LMI drives DQS pins in write (Store) cycles, while DDR SDRAM drives it in read (Load) cycles.

Table 5: SDRAM interface pins

Name	I/O	Size	Description
DQM[7:0]	Output	8	Input/output data mask For regular SDRAM, these pins act as synchronized output enables during read cycles and as byte enables during write cycles. For DDR SDRAM, these pins act as byte enables during write cycles.
NOT_RASA NOT_RASB	Output	1 1	Row address strobe The NOT_RASA and NOT_RASB are multiple copies of the same logic RAS signal used to generate encoded SDRAM command.
NOT_CASA NOT_CASB	Output	1 1	Column address strobe The NOT_CASA and NOT_CASB are multiple copies of the same logic CAS signal used to generate encoded SDRAM command.
VREF	Input	1	Input reference voltage

Table 5: SDRAM interface pins

A. There are two copies of NOT_CS for each physical memory row to reduce the loading.

To accommodate various loading conditions, the buffer strength of the pins is programmable. This feature can minimize unnecessary power consumption while still meeting the SDRAM device’s timing requirements. See *Section 1.4.5* for details.

1.3.3 SDRAM devices

The LMI splits the physical memory address into banks, (internal) row and column addresses. The LMI contains 17 external address pins. BA[1:0] specifies which bank, while MA[14:0] indicates row and column addresses in each bank. The (internal) row address selects a page in an SDRAM. The column address selects a datum in a row. The LMI supports memories where row addresses are up to 15 bits.

The following three tables summarize various SDRAM devices which are used to construct the memory subsystem in three different data bus widths. They also illustrate MA pins mux-ing vs SDRAM address split. LMI’s MA[14:0] pins are directly connected to SDRAM’s A[14:0]. The address split column in the table specifies the row and column address split within a given bank.

Using the second entry as an example, 2 of 16 Mbytes (2Mx8 type, two banks) SDRAMs are used to construct a row of main memory. The SDRAM’s internal row and column address bits are 11 and 9, respectively. The page size is 1 Kbyte. Total memory on this row is 4 Mbytes. The CPU’s physical address PA [11] is output to BA[0] pins in both RAS and CAS phases. MA[10] is driven with PA [12] in RAS phase. AP (Auto Precharge) option is output to MA[10] in CAS phase, although the ST40 LMI does not issue either read-with auto-precharge or write-with auto-precharge commands.

16-bit data bus interface

SDRAM type	Address split	Page size	Row size	RAS CAS	BA1	BA0	MA12	MA11	MA10 /AP	MA9	MA8	MA [7:0]
16 Mbit 2 bank												
1 Mbit x 16	11 x 8	512 bytes	2 Mbytes	RAS CAS		11 11			12 AP	10	9	[20:13] [8:1]
2 Mbit x 8	11 x 9	1 Kbytes	4 Mbytes	RAS CAS		11 11			12 AP	10	21 9	[20:13] [8:1]
4 Mbit x 4	11 x 10	2 Kbytes	8 Mbytes	RAS CAS		11 11			12 AP	22 10	21 9	[20:13] [8:1]
64 Mbit 2 bank												
4 Mbit x 16	13 x 8	512 bytes	8 Mbytes	RAS CAS		11 11	12	10	9 AP	22	21	[20:13] [8:1]
8 Mbit x 8	13 x 9	1 Kbytes	16 Mbytes	RAS CAS		11 11	12	10	23 AP	22	21 9	[20:13] [8:1]
16 Mbit x 4	13 x 10	2 Kbytes	32 Mbytes	RAS CAS		11 11	12	24	23 AP	22 10	21 9	[20:13] [8:1]
64 Mbit 4 bank												
4 Mbit x 16	12 x 8	512 bytes	8 Mbytes	RAS CAS	12 12	11 11		10	9 AP	22	21	[20:13] [8:1]
8 M bit x 8	12 x 9	1 Kbyte	16 Mbytes	RAS CAS	12 12	11 11		10	23 AP	22	21 9	[20:13] [8:1]
16 Mbit x 4	12 x 10	2 Kbytes	32 Mbytes	RAS CAS	12 12	11 11		24	23 AP	22 10	21 9	[20:13] [8:1]
128 Mbit 4 bank												
8 Mbit x 16	12 x 9	1 Kbyte	16 Mbytes	RAS CAS	12 12	11 11		10	23 AP	22	21 9	[20:13] [8:1]
16 Mbit x 8	12 x 10	2 Kbytes	32 Mbytes	RAS CAS	12 12	11 11		24	23 AP	22 10	21 9	[20:13] [8:1]
32 Mbit x 4	12 x 11	4 Kbytes	64 Mbytes	RAS CAS	12 12	25 25		24 11	23 AP	22 10	21 9	[20:13] [8:1]

Table 6: Row and column addressing for memory size and number of banks (32-bit interface)

SDRAM type	Address split	Page size	Row size	RAS CAS	BA1	BA0	MA12	MA11	MA10 /AP	MA9	MA8	MA [7:0]
256 Mbit 4 bank												
16 Mbit x 16	13 x 9	1 Kbyte	32 Mbytes	RAS CAS	12 12	11 11	10	24	23 AP	22	21 9	[20:13] [8:1]
32 Mbit x 8	13 x 10	2 Kbytes	64 Mbytes	RAS CAS	12 12	11 11	25	24	23 AP	22 10	21 9	[20:13] [8:1]
64 Mbit x 4	13 x 11	4 Kbytes	128 Mbytes	RAS CAS	12 12	26 26	25	24 11	23 AP	22 10	21 9	[20:13] [8:1]

Table 6: Row and column addressing for memory size and number of banks (32-bit interface)

32-bit data bus interface

SDRAM type	Address split	Page size	Row size	RAS CAS	BA1	BA0	MA12	MA11	MA10 /AP	MA9	MA8	MA [7:0]
16 Mbit 2 bank												
1Mbit x 16	11 x 8	1 Kbytes	4 Mbytes	RAS CAS		13 13			12 AP	11	10	[21:14] [9:2]
2Mbits x 8	11 x 9	2 Kbytes	8 Mbytes	RAS CAS		13 13			12 AP	11	22 10	[21:14] [9:2]
4Mbits x 4	11 x 10	4 Kbytes	16 Mbytes	RAS CAS		13 13			12 AP	23 11	22 10	[21:14] [9:2]
64Mbit 2 bank												
4Mbits x 16	13 x 8	1 Kbytes	16 Mbytes	RAS CAS		13 13	12	11	10 AP	23	22	[21:14] [9:2]
8Mbits x 8	13 x 9	2 Kbytes	32 Mbytes	RAS CAS		13 13	12	11	24 AP	23	22 10	[21:14] [9:2]
16Mbits x 4	13 x 10	4 Kbytes	64 Mbytes	RAS CAS		13 13	12	25	24 AP	23 11	22 10	[21:14] [9:2]
64Mbit 4 bank												
2Mbits x 32	11 x 8	1 Kbytes	8 Mbytes	RAS CAS	12 12	13 13			10 AP	11	22 AP*	[21:14] [9:2]
4Mbits x 16	12 x 8	1 Kbytes	16 Mbytes	RAS CAS	12 12	13 13		11	10 AP	23	22	[21:14] [9:2]
8Mbits x 8	12 x 9	2 Kbytes	32 Mbytes	RAS CAS	12 12	13 13		11	24 AP	23	22 10	[21:14] [9:2]
16Mbits x 4	12 x 10	4 Kbytes	64 Mbytes	RAS CAS	12 12	13 13		25	24 AP	23 11	22 10	[21:14] [9:2]
128 Mbit 4 bank												
8 Mbit x 16	12 x 9	2 Kbytes	32 Mbytes	RAS CAS	12 12	13 13		11	24 AP	23	22 10	[21:14] [9:2]
16 Mbit x 8	12 x 10	4 Kbytes	64 Mbytes	RAS CAS	12 12	13 13		25	24 AP	23 11	22 10	[21:14] [9:2]

Table 7: Row and column addressing for memory size and number of banks (32-bit interface)

SDRAM type	Address split	Page size	Row size	RAS CAS	BA1	BA0	MA12	MA11	MA10 /AP	MA9	MA8	MA [7:0]
32 Mbit x 4	12 x 11	8 Kbytes	128 Mbytes	RAS CAS	26 26	13 13		25 12	24 AP	23 11	22 10	[21:14] [9:2]
256 Mbit 4 bank												
16 Mbit x 16	13 x 9	2 Kbytes	64 Mbytes	RAS CAS	12 12	13 13	11	25	24 AP	23	22 10	[21:14] [9:2]
32 Mbit x 8	13 x 10	4 Kbytes	128 Mbytes	RAS CAS	12 12	13 13	26	25	24 AP	23 11	22 10	[21:14] [9:2]
64 Mbit x 4	13 x 11	8 Kbytes	256 Mbytes	RAS CAS	27 27	13 13	26	25 12	24 AP	23 11	22 10	[21:14] [9:2]

Table 7: Row and column addressing for memory size and number of banks (32-bit interface)

64-bit data bus interface

SDRAM type	Address split	Page size	Row size	RAS CAS	BA 1	BA0	MA12	MA11	MA10 /AP	MA9	MA8	MA [7:0]
16 Mbit 2 bank												
1 Mbit x 16	11 x 8	2 Kbytes	8 Mbytes	RAS CAS		13 13			14 AP	12	11	[22:15] [10:3]
2 Mbits x 8	11 x 9	4 Kbytes	16 Mbytes	RAS CAS		13 13			14 AP	12	23 11	[22:15] [10:3]
4 Mbits x 4	11 x 10	8 Kbytes	32 Mbytes	RAS CAS		13 13			14 AP	24 12	23 11	[22:15] [10:3]
64 Mbit 2 bank												
4 Mbits x 16	13 x 8	2 Kbytes	32 Mbytes	RAS CAS		13 13	14	12	11 AP	24	23	[22:15] [10:3]
8 Mbits x 8	13 x 9	4 Kbytes	64 Mbytes	RAS CAS		13 13	14	12	25 AP	24	23 11	[22:15] [10:3]
16 Mbits x 4	13 x 10	8 Kbytes	126 Mbytes	RAS CAS		13 13	14	26	25 AP	24 12	23 11	[22:15] [10:3]
64 Mbit 4 bank												
2 Mbits x 32	11 x 8	2 Kbytes	16 Mbytes	RAS CAS	14 14	13 13			11 AP	12	23 AP*	[22:15] [10:3]
4 Mbits x 16	12 x 8	2 Kbytes	32 Mbytes	RAS CAS	14 14	13 13		12	11 AP	24	23	[22:15] [10:3]
8 Mbits x 8	12 x 9	4 Kbytes	64 Mbytes	RAS CAS	14 14	13 13		12	25 AP	24	23 11	[22:15] [10:3]
16 Mbits x 4	12x10	8K	128M	RAS CAS	14 14	13 13		26	25 AP	24 12	23 11	[22:15] [10:3]
128 Mbit 4 bank												
8 Mbit x 16	12 x 9	4 Kbytes	64 Mbytes	RAS CAS	14 14	13 13		12	25 AP	24	23 11	[22:15] [10:3]
16 Mbit x 8	12 x 10	8 Kbytes	128 Mbytes	RAS CAS	14 14	13 13		26	25 AP	24 12	23 11	[22:15] [10:3]

Table 8: Row and column addressing for memory size and number of banks (64-bit interface)

SDRAM type	Address split	Page size	Row size	RAS CAS	BA 1	BA0	MA12	MA11	MA10 /AP	MA9	MA8	MA [7:0]
32 Mbit x 4	12 x 11	16 Kbytes	256 Mbytes	RAS CAS	14 14	27 27		26 13	25 AP	24 12	23 11	[22:15] [10:3]
256 Mbit 4 bank												
16 Mbit x 16	13 x 9	4 Kbytes	128 Mbytes	RAS CAS	14 14	13 13	12	26	25 AP	24	23 11	[22:15] [10:3]
32 Mbit x 8	13 x 10	8 Kbytes	256 Mbytes	RAS CAS	14 14	13 13	27	26	25 AP	24 12	23 11	[22:15] [10:3]
64 Mbit x 4	13 x 11	16 Kbytes	512 Mbytes	RAS CAS	14 14	28 28	27	26 13	25 AP	24 12	23 11	[22:15] [10:3]

Table 8: Row and column addressing for memory size and number of banks (64-bit interface)

Note: AP PIN: *LMI uses the* MIM.BY32AP *bit to determine if the* MA8 *pin is used to indicate the PRE and PALL commands.*

1.3.4 Initializing SDRAM devices

An initialization sequence to an SDRAM device must be done after power-on reset by the driver software. The operating system boot-up code or driver software to initialize SDRAM should not read or write SDRAM before completion of the initialization.

Single data rate SDRAM

1 VDD and VDDQ are applied simultaneously on power-up by the system to meet the SDRAM specification.

2 CKE is initially low after a power-on reset. With the CKE enable command, write to the LMI.SCR register and all CKE becomes high.

3 A minimum pause of 200 µs (some of SDRAM only require 100 µs) needs to be provided after powers are stable.

4 A precharge all (PALL) is issued to SDRAM.

5 Eight autorefresh commands, that is CAS-before-RAS (CBR) cycles send to SDRAM.

6 A mode register set (MRS) command is issued to program the SDRAM parameters, for example burst length, and CAS_ Latency.

7 A legal command for normal operation can be started after an SDRAM-specific AC timing.

8 From reset to the completion of mode register set, the LMI should not drive DQ to avoid contention of drivers. At the same time DQM should keep high.

Mode register set and 8 CBR can be transposed. The minimum pause time might vary from the generation and the vendor of SDRAM. This LMI should be able to support both 100 µs or 200 µs providing the pause period by software allows this.

Double data rate SDRAM

1 The system provides four power in certain sequence. VDD first, VDDQ next then VREF and VTT. VTT is not provided to the LMI, it is externally connected to DQ, DQS and other pins through a series of termination registers. This is required to prevent latch-up in SDRAM devices. LMI should be able to support this power-up sequence. During and after power-on reset, CKE must be kept low.

2 After all power supply and reference voltages are stable, and the clock is stable a 200 µs pause is necessary.

3 CKE should be brought high with the DESELECT command. After this point, unless LMI sends some command, LMI has to send the DESELECT command.

4 A precharge all (PALL) is issued to SDRAM.

5 A mode register set (MRS) command is issued to program the extended mode register to enable the DLL.

6 The MRS command is issued to program the mode register, reset the DLL in SDRAM and program the operating parameters, for example burst length and CAS_LATENCY.

7 Wait ten cycles after the DLL reset and send two CBR commands to SDRAM.

8 A MRS command is issued to de-assert DLL initialization bit in the mode register. Other programing parameters should be the same as in previous programing. For some memory vendors, this step can be skipped because they support auto cleaning of the DLL initialization bit.

9 After 200 cycles from DLL reset, external memory becomes accessible.

The LMI's SDRAM controller provides two mechanisms for accomplishing the initialization sequence.

1 NOP, PALL, CKEH and CBR

   The SCR register's SMS (SDRAM mode select) field is written with appropriate values to prompt the SDRAM controller to start issuing to start issuing one of these commands. For instance, when SCR.SMS = 3'b100, it results in a single CBR cycle on the SDRAM interface. When SMS = 3'b011, it results in the CKE signals going high. See *Section 1.4.3: SDRAM control register (LMI.SCR) on page 40* for details.

2 Setting the SDRAM device's mode register.Mode Register

   The SDRAM's mode register needs to be initialized before actual operation. The software (boot code) initiates a write cycle to the MIM register, and then a write to the SDMR[N] register in the control block. The SDRAM controller then issues an MRS command to all SDRAM devices on row (row [n]).

Example: issuing MRS command to row 0

Software does a dummy write to SDMR0, the physical address must be arranged in the following way:

- A[31:20]: 1111 1111 1000r,
- A[16:3] contains the value to be written to the SDRAM's mode register,
- data [63:0] is ignored since SDMR0 is a write-only virtual register,
- A[12:3] is copied to MA[9:0], A[18:15] to MA[13:10] and A [14:13] to BA[1:0] when an MRS command is issued to the SDRAM devices.

Software needs to ensure that the SDRAM timing specification (between the MRS command and the first operational command) is met. one way to ensure this is to perform several SDRAM control register reads.

Subsequently, SCR.SMS is written with 3'b000, the normal SDRAM operation can then be started.

Note: *Software must program the LMI's* MIM *register before writing to* SDMR[N].

1.3.5 Operations

The SDRAM controller supports most PC-SDRAM commands (with the exception of read/write with auto-precharge) and most DDR SDRAM commands. The following truth table lists up all commands supported.

Function	Symbol	CKE [n - 1]	CKE [n]	NOT _CS	NOT _RAS	NOT _CAS	NOT _WE	MA11	AP[A] (MA10 /MA8)	BA [1:0]	MA [9:0]
Device deselect	DSEL	H	X	H	X	X	X	X	X	X	X
No operation	NOP	H	X	L	H	H	H	X	X	X	X
Burst stop in read	BST	H	X	L	H	H	L	X	X	X	X
Read	READ	H	X	L	H	L	H	V	L	V	V
Write	WRITE	H	X	L	H	L	L	V	L	V	V
Bank activate	ACT	H	X	L	L	H	H	V	V	V	V
Precharge select bank	PRE	H	X	L	L	H	L	V	L	V	X
Precharge all banks	PALL	H	X	L	L	H	L	X	H	X	X
Auto refresh	CBR	H	H	L	L	L	H	X	X	X	X
Self refresh entry from idle	SLFRSH	H	L	L	L	L	H	X	X	X	X
Self refresh exit	SLFRSHX	L	H	H	X	X	X	X	X	X	X
Power-down entry from idle	PWRDN	H	L	X	X	X	X	X	X	X	X
Power-down exit	PWRDNX	L	H	H	X	X	X	X	X	X	X
Mode register set	MRS	H	X	L	L	L	L	V	V	V	V

Table 9: SDRAM command truth table

A. AP pin: LMI uses MIM.BY32AP bit to determine if MA8 pin is used to indicate PRE and PALL commands.

Note: The LMI does not support full-page burst operation. The LMI issues a BST *command to terminate the burst read-only in DDR SDRAM mode.*

The timing for issuing these commands is governed by the SDRAM timing register, see *Section 1.4.4: SDRAM timing register (LMI.STR) on page 43* for details. The LMI's SDRAM controller can open up to four pages for each SDRAM row and fully exploit the multi-bank architecture of modern SDRAM devices by tightly pipelining SDRAM commands. The LMI is capable of detecting multiple consecutive requests to the same SDRAM page. The SDRAM controller may combine same-page requests into a single same-page access, providing that the timing of the requests is suitable.

Multi-bank ping-pong transaction

Two bank ping-pong access is illustrated in the following diagram. The peak bandwidth is obtained in this scenario.

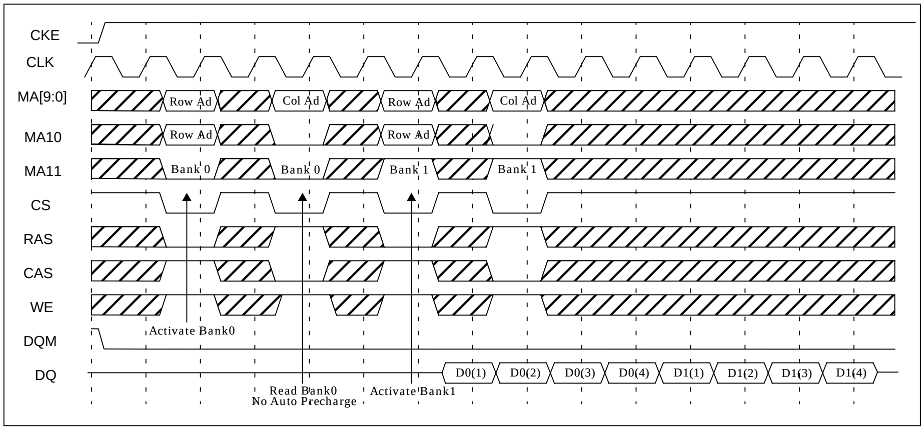


Figure 2: PC SDRAM two bank ping pong read

1.3.6 Refresh

When DRAM refresh enable is 1 (MIM.DRE = 1), The LMI can automatically generate refresh cycles. A 12-bit quantity (MIM.DRI, DRAM refresh interval) specifies the number of memory clock cycles between refreshes. Software should program MIM.DRI in the inclusive range [128:4095]. The behavior of the DRAM controller is undefined if the LMI is enabled and if DRI is less than 128.

At the start of a refresh interval, the SDRAM controller loads DRI's 12-bit value into an internal counter. This counter is decremented by 1 in each memory clock cycle.

When the counter reaches 0, DRI's value is reloaded into the counter and the next refresh interval is started.

All banks must be closed before refresh operation can be performed. The SDRAM controller issues a PRECHARGE ALL (PALL) command if there are any open pages. The SDRAM controller then issues an AUTO REFRESH command (CBR) after the TRP parameter is satisfied. The next row ACT command can be issued Trc clock (LMI.STR.SRC) later.

The SDRAM controller performs exactly one refresh operation for each refresh interval. It attempts to perform CBR as soon as possible within the refresh interval. When the counter ≤ 128 and CBR is not issued in the current refresh interval the SDRAM controller causes any current SDRAM access to complete in a timely manner by ensuring that the detection of same-page SDRAM access is prevented. Subsequently it performs PALL and CBR commands.

The maximum refresh rate that the LMI can support is one row every 128 clock cycles. At this rate, however, the detection of same-page SDRAM accesses will be permanently disabled.

As an example, the hard reset value of DRI is 1,562. For 100 MHz MCLKO, then this allows 1,024 refreshes in less than 16 ms.

Note: *On average, the interval between two refreshes is determined by the DRI setting. However the interval between any two successive refreshes could be larger or smaller than DRI by (a page miss 32-byte transfer) clocks.*

1.3.7 Power management

The LMI provides one power management mechanism.

When the LMI receives STBY_REQ from the power-down management unit (PMU), the LMI prepares the SDRAM rows to enter low power state. The sequence of events for both entering and leaving standby mode is described below. To make the correct sequence, cooperation with the software driver is important.

Entering standby

1 At first, no initiators should be issuing transaction requests to the LMI.

2 The standby management program should issue CBR command as the last command to LMI.

3 The standby management program asserts STBY_REQ to LMI.

4 All outstanding transaction requests are serviced.

5 The SDRAM controller issues a self refresh command and lowers CKE[1:0] to both SDRAM rows. The SDRAM autonomously refreshes itself for the duration of the power-down mode.

6 LMI asserts STBY_ACK to PMU. The clock (MCLKO) can now be stopped.

Leaving standby by causes other than power-on reset

1 PMU resumes the LMI's SuperHyway clock and SDRAM clock and deasserts STBY_REQ.

2 The LMI de-asserts STBY_ACK, and starts to count down from (256 x SCR.CST) to zero every MCLK cycle.

3 When count down reaches zero, the SDRAM controller asserts all CKE[1:0] pins and sends **deselect** commands continuously. All SDRAM rows exit from self-refresh mode.

4 The first valid command can be issued ten cycles after CKE's rising edge.

5 In the case of DDR SDRAM, the LMI issues numbers of CBR commands defined by the CSR.BRFSH field.

1.3.8 Caution when programming SDRAM’s mode register

To effectively support SuperHyway Load32 and Store32 packets, the LMI’s SDRAM controller uses MIM.DT (SDRAM type) and MIM.BW (external data bus width) to determine the burst length.

MIM.DT device type	MIM.BW bus width	Burst length
0: PC SDRAM	00: 16-bit	8
	01: 32-bit	8
	10: 64-bit	4
1: DDR SDRAM	00: 16-bit	8
	01: 32-bit	8
	10: 64-bit	4

Table 10: Determining burst length for Load32 and Store32 packets

For a 16-bit external data bus width (for either PC-SDRAM or DDR SDRAM), the LMI splits a SuperHyway Load32 or Store32 packet into multiple SDRAM transactions, with a burst of eight for each transaction. Therefore the BL field of the SDRAM device’s mode register must be programmed to match the LMI’s burst length behavior in the third column.

1.3.9 Using registered DIMM

When using registered DIMM, the MIM register’s DIMM bit needs to be set to 1, so that LMI can:

- delay data output by one cycle to synchronize with the buffered (on DIMM card) command signals before they reach the SDRAM devices during a write operation,
- add one MCLK cycle to the setting of STR.SCL bit (CAS Latency). STR.SCL bits should be programmed with the same CL latency as the CL setting in the SDRAM device’s mode register.

1.3.10 Others

Memory access to the address range of the base address + (0x08000000 to 0xEFFFFFF), is routed to the LMI. This address range may not be fully populated with the SDRAMs. Data access beyond the populated address, as defined in SDRA1's UBA, will not result in an external memory transaction. Software that dynamically sizes the amount of external memory must use an algorithm that is aware of this property. In the case of DIMMs, software can use I/O pins to implement a serial presence detect (SPD) mechanism for dynamic sizing of main memory.

1.4 Register description

1.4.1 Version control register (LMI.VCR)

VCR is defined in the *System Architecture Manual, Volume 1, Appendix A: ST40 System Architectural Conventions*. The control block of every SuperHyway module contains this register.

LMI.VCR				0x0F00 0000	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PERR	[0:7]	8	Yes	Port error flags	Vary
	Operation		Indicates errors in the interface between LMI and the packet-router		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
MERR	[8:15]	8	Yes	Memory error flags	Vary
	Operation		Indicates errors in the LMI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
MVERS	[16:31]	16	No	Module version	RO
	Operation		Indicates module version number		
	Read		Returns 0x0001		
	Write		Ignored		
	Hard reset		0x0001		

Table 11: LMI.VCR

LMI.VCR				0x0F00 0000	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MID	[32:47]	16	No	Module identity	RO
	Operation		Identifies module		
	Read		Returns 0x1000		
	Write		Ignored		
	Hard reset		0x1000		
BOT_MB	[48:55]	8	No	Bottom data memory block	RO
	Operation		Identifies top of data memory block		
	Read		0x080		
	Write		Ignored		
	Hard reset		0x080		
TOP_MB	[56:63]	8	No	Top data memory block	RO
	Operation		Used to identify top of data memory block		
	Read		In ST40 it returns 0x0F		
	Write		Ignored		
	Hard reset		0x0F		

Table 11: LMI.VCR

The error status due to access to the LMI's control block is recorded in the PERR field. The set of supported PERR flags in LMI.VCR is given in the following table. The bit positions in this table are relative to the start of the LMI.VCR.PERR field. This field starts at bit 0 of LMI.VCR.

LMI.VCR.PERR			0x0F00 0000		
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	0	1	Yes	Reserved	Res
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		
ERR_SNT	1	1	Yes	Error response sent	RW
	Operation		Indicates an error response has been sent This bit is set by the LMI hardware if an error response is sent by the LMI to SuperHyway. It indicates that an earlier request to the LMI was invalid		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
BAD_ADDR	2	1	Yes	Undefined control register	RW
	Operation		Indicates a request for an undefined control register has been received This bit is set by the LMI hardware if the LMI hardware receives a request for an undefined control register.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		

Table 12: LMI.VCR.PERR

LMI.VCR.PERR			0x0F00 0000		
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[3:4]	2	Yes	Reserved	Res
	Operation				
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		
BAD_OPC	5	1	Yes	Unsupported op code	RW
	Operation		Indicates a request with an unsupported opcode has been received This bit is set by the LMI hardware if a request with an unsupported opcode is received by LMI from SuperHyway.		
	Read		Returns current value		
	Write		Update current value		
	Hard reset		0		
-	[6:7]	2	Yes	Reserved	Res
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 12: LMI.VCR.PERR

The error status due to access to the LMI’s data block is recorded in MERR field. The set of supported MERR flags in LMI.VCR is given in *Table 13*. The bit positions in this table are relative to the start of the LMI.VCR.MERR field. This field starts at bit 8 of LMI.VCR.

LMI.VCR.MERR					
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
DRAM_INACTIVE	0	1	Yes	Access to LMI data block (i.e. external memory) when DRAM controller is disabled	RW
	Operation		This bit is set by the LMI hardware if a request is made to access external memory while DRAM controller is disabled.		
	Read		Returns current value		
	Write		Update current value		
	Hard reset		0		
ERR_SNT	1	1	Yes	An error response has been sent	RW
	Operation		This bit is set by the LMI hardware if an error response is sent by the LMI to SuperHyway. It indicates that an earlier request to the LMI's data block was invalid		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
BAD_ADDR	2	1	Yes	A request to an out-of-range or unpopulated address has been received	RW
	Operation		This bit is set by the LMI hardware if the LMI hardware receives a request directed to an out-of-range address or an unpopulated address in data block		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		

Table 13: LMI.VCR.MERR

LMI.VCR.MERR					
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[3:4]	2	Yes	Reserved	Res
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		
BAD_OPC	5	1	Yes	A request with an unsupported opcode has been received	RW
	Operation		This bit is set by the LMI hardware if a request with an unsupported opcode is received by LMI from SuperHyway		
	Read		Returns current value		
	Write		Update current value		
	Hard reset		0		
-	[6:7]	2	Yes	Reserved	Res
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 13: LMI.VCR.MERR

1.4.2 Memory interface mode register (LMI.MIM)

LMI.MIM register specifies the configuration of the DRAM interface.

LMI.MIM				**0x0F000008**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
DCE	0	1	No	DRAM controller enable	RW
	Operation		Indicates whether the SDRAM controller is enabled or disabled When the SDRAM controller is disabled, the LMI generates error responses to SuperHyway requests (for data block access) directed to the LMI.		
	Read		Returns current value		
	Write		0: SDRAM controller is disabled 1: SDRAM controller is enabled		
	Hard reset		0		
DT	1	1	No	DRAM type	RO
	Operation		Specifies the SDRAM type		
	Read		0: PC-SDRAM 1: DDR SDRAM		
	Write		Updates current value		
	Hard reset		Undefined		
-	[2:5]	4	No	Reserved	Res
	Operation		Reserved		
	Read		Ignored		
	Write		0		
	Hard reset		Undefined		

Table 14: LMI.MIM

LMI.MIM				0x0F000008	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
BW	[6:7]	2	No	Bus width	RW
	Operation		Indicates the data bus width of the LMI's SDRAM interface		
	Read		Returns current value		
	Write		00: 16 01: 32 10: 64 11: Reserved		
	Hard reset		10		
ENDIAN	8	1	No	Memory endianness	RO
	Operation		Indicates whether the memory configuration is little or big endian This bit only affects 16-bit and 32-bit bus width modes. For 64-bit external interface, the endianness is transparent to the LMI.		
	Read		0: Little endian memory configuration 1: Big endian memory configuration		
	Write		Ignored		
	Hard reset		0		
DRE	9	1	No	DRAM refresh enable	RW
	Operation		Enable refresh mechanism		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		

Table 14: LMI.MIM

LMI.MIM				0x0F000008	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
DIMM	10	1	No	Registered DIMM module	RW
	Operation		Constructs the external row		
	Read		Returns current value		
	Write		1: Data output delayed by one MCLK cycle and one MCLK cycle added to CAS latency		
	Hard reset		0		
BY32AP	11	1	No	Interfacing x32 SDRAM devices	RW
	Operation		Pre-charges all bank command’s indicator for x32 SDRAM devices		
	Read		Returns current value		
	Write		0: BY32AP MA8 pin is not used when LMI issues PRE or PALL commands. 1: BY32AP MA8 pin is used when LMI issues PRE or PALL commands.		
	Hard reset		0		
-	[12:15]	4	No	Reserved	Res
	Operation		Reserved		
	Read		Ignored		
	Write		0		
	Hard reset		Undefined		

Table 14: LMI.MIM

LMI.MIM				0x0F000008	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
DRI	[16:27]	12	No	DRAM refresh interval	RW
	Operation		Determines the maximum number of memory clock cycles between row refreshes, when enabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0x61A		
-	[28:63]	36	No	Reserved	Res
	Operation		Reserved		
	Read		Ignored		
	Write		0		
	Hard reset		Undefined		

Table 14: LMI.MIM

1.4.3 SDRAM control register (LMI.SCR)

LMI.SCR				0x0F000010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
SMS	[0:2]	3	No	SDRAM mode select	RW
	Operation		Enables the SDRAM controller to perform normal SDRAM operation and to issue NOP, PALL and CBR which are required in the SDRAM device initialization sequence for power up or reset		
	Read		Returns current value		
	Write		000: Normal SDRAM operation when MIM.DCE = 1 001: NOP command enable When SMS is written with this value and MIM.DCE = 1, the LMI issues one NOP command to the SDRAM interface. To have [n] number of NOP commands, SCR.SMS must be written with 001 [n] times. This command applies to all external SDRAM rows 010: Precharge all banks When SMS is written with this value and MIM.DCE = 1, the LMI issues one PRECHARGE ALL command to the SDRAM interface. To have [n] number of PALL commands, SCR.SMS must be written with 010 [n] times. This command applies to all external SDRAM rows. 011: Clock enable signals LCLKEN0 and LCKLKEN1 active At reset the clocks are disabled. 100: CBR enable When SMS is written with this value and MIM.DCE = 1, the LMI issues one CBR command to the SDRAM interface. To have [n] number of CBR commands, SCR.SMS must be written with 011 [n] times. This command applies to all external SDRAM rows. 101, 110, 111: Reserved		
	Hard reset		3’b000		

Table 15: LMI.SCR

LMI.SCR				0x0F000010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PDSE	3	1	No	Power-down SDRAM enable	RW
	Operation		Allows the SDRAM controller to issue a power-down command to an idle SDRAM row The SDRAM controller issues a power down exit command when there is a request to access this idle row.		
	Read		Returns current value		
	Write		0: Disable 1: Enable		
	Hard reset		0		
BRFSH	[4:6]	3	no	Burst Refresh	RW
	Operation		This bit enables burst refresh after wake up from standby. 000: no. 001: 32 010: 512 011: 1024 100: 2048 101: 4096 110: reserved 111: reserved		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		

Table 15: LMI.SCR

LMI.SCR				0x0F000010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
CST	[16 - 27]	12	no	Clock Stabilization Time	RW
	Operation		These bits specify clock stabilization time on return from module standby mode. 0: There is no stabilization time before LMI start operation with SDRAM. <>0: Count down this number with 1/256 MCLK. When it reached to 0, LMI starts operation.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
-	3, [7 - 15], [28 - 63]	49	No	Reserved	Res
	Operation		Reserved		
	Read		Ignored		
	Write		0		
	Hard reset		Undefined		

Table 15: LMI.SCR

1.4.4 SDRAM timing register (LMI.STR)

LMI.STR				0x0F000018	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
SRP	0	1	No	Trp, RAS precharge to ACT command	RW
	Operation		Controls the number of MCLKS for RAS precharge to ACT command (for the same bank)		
	Read		Returns current value		
	Write		0: two clocks of RAS precharge 1: three clocks of RAS precharge		
	Hard reset		Undefined		
SRCD	1	1	No	TRCD, RAS to CAS delay	RW
	Operation		Controls the number of MCLKS from a ROW ACTIVATE command to a column command (for the same bank)		
	Read		Returns current value		
	Write		0: two clocks of RAS to CAS delay. 1: three clocks of RAS to CAS delay		
	Hard reset		Undefined		

Table 16: LMI.STR

LMI.STR				0x0F000018	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
SCL	[2:4]	3	No	SDRAM CAS latency (CL)	RW
	Operation		Controls the number of MCLKS between when a read command is sampled by the SDRAMs and when the processor samples read data from SDRAMs		
	Read		Returns current value		
	Write		010: 2 clocks 011: 3 clocks 101: 1.5 clocks 110: 2.5 clocks All others = Reserved		
	Hard reset		Undefined		
SRC	[5:7]	3	No	Trc, RAS cycle time.	RW
	Operation		Minimum delay between ACT and Auto Refresh (to the same bank) ACT and ACT (to the same bank) Auto Refresh and ACT (to the same bank) Auto Refresh and Auto Refresh (to the same bank)		
	Read		Returns current value		
	Write		000: Six clocks 001: Seven clocks 010: Eight clocks 011: Nine clocks All others = Reserved		
	Hard reset		Undefined		

Table 16: LMI.STR

LMI.STR				0x0F000018	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
SRAS	[8:10]	3	No	Tras, RAS active time	RW
	Operation		ACT to PRE command (for the same bank)		
	Read		Returns current value		
	Write		001: Five clocks 010: Six clocks 011: Seven clocks 100: Eight clocks 101: Nine clocks All others = Reserved		
	Hard reset		Undefined		
SRRD	11	1	No	Trrd, RAS to RAS active delay	RW
	Operation		Specifies delay from ACT bank [n] to ACT bank [i] command (different bank)		
	Read		Returns current value		
	Write		0: two clocks 1: three clocks		
	Hard reset		Undefined		
SDPL	12	1	No	SDRAM Tdpl, as well as DDR SDRAM's Twr	RW
	Operation		SDRAM: last write-data to PRE or PALL command period DDR SDRAM: from the end of postamble to PRE or PALL command		
	Read		Returns current value		
	Write		0: one clock 1: two clocks		
	Hard reset		Undefined		

Table 16: LMI.STR

LMI.STR				0x0F000018	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[13:63]	51	No	Reserved	Res
	Operation		Reserved		
	Read		Ignored		
	Write		0		
	Hard reset		Undefined		

Table 16: LMI.STR

1.4.5 SDRAM row attribute registers (LMI.SDRA[0:1])

LMI.SDRA[0:1]				**0x0F000030, 0x0F000038**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[0:7]	8	No	Reserved	Res
	Operation		Reserved		
	Read		Ignored		
	Write		0		
	Hard reset		Undefined		
SPLIT	[8:11]	4	No	SDRAM device address split for each bank	RW
	Operation		Defines the split of row and column address bits for a given bank within an SDRAM device		
	Read		Returns current value		
	Write		0000: 11x8 0001: 11x9 0010: 11x10 0011: Reserved 0100: 12x8 0101: 12x9 0110: 12x10 0111: Reserved 1000: 13x8 1001: 13x9 1010: 13x10 1011: 13x11 11nn = Reserved		
	Hard reset		Undefined		

Table 17: LMI.SDRA[0:1]

LMI.SDRA[0:1]				0x0F000030, 0x0F000038	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
BANK	12	1	No	SDRAM device bank number	RW
	Operation		Defines the SDRAM device bank number of the associated physical memory row		
	Read		Returns current value		
	Write		0: Dual-bank 1: Quad-bank		
	Hard reset		Undefined		
-	[13:20]	8	No	Reserved	Res
	Operation		Reserved		
	Read		Ignored		
	Write		0		
	Hard reset		Undefined		
UBA	[21:31]	11	No	Row upper boundary address	RW
	Operation		Defines the upper boundary address of the external SDRAM row in 2 Mbytes granularity UBA specifies the external row’s upper boundary address [21:31].		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		Undefined		
-	[32:63]	32	No	Reserved	Res
	Operation		Reserved		
	Read		Ignored		
	Write		0		
	Hard reset		Undefined		

Table 17: LMI.SDRA[0:1]

1.4.6 SDRAM row mode registers (LMI.SDMR[0:1])

These registers are write-only virtual registers, since physically they are not contained in the processor chip. A write to these virtual registers triggers an SDRAM mode register set command to be issued to a row of SDRAM devices. The value on physical address A[12:3] is copied to MA[9:0] pins, A[14:13] is output to BA[1:0] and A[18:15] is driven to MA[13:10]. The values on data pins are undefined and are ignored by the SDRAM devices. In response to the mode register set command, an SDRAM or DDR SDRAM device then latches MA[11:0] and BA[1:0] into its mode register. A read to these registers returns undefined value.

Please refer to the SDRAM and DDR SDRAM manufacture's data sheets for the definition of each bit in its mode register and extended mode register.

Note: The definition of a mode register's bit field varies with different SDRAM density.

1.5 References

HM54S64XXX series 64M LVTLL interface SDRAM data sheet

HM54S64XXD2 series double data rate SDRAM data sheet

Intel PC100 SDRAM specification

2 Enhanced flash memory interface (EMI)[1]

2.1 Overview

The enhanced flash memory interface (EMI) is a general purpose external memory interface which allows the system to support a number of memory types, external process interfaces and devices. This includes glue-less support for up to six independent memories or devices. The EMI allows external devices to become master of the memory bus to support features such as external DMAs and bus mastering.

2.2 Features

The main features include:

- up to 100 MHz operating frequency,
- SuperHyway transaction support,
- support for up to six external memory banks,
- SDRAM support with possible subdecoding (connectable banks 0 to 5),
- burst flash support (AMD AM29BL162C, ST M58LW064, Intel 28F160F3 compatible) (connectable banks 0 to 5),
- MPX initiator only support (connectable banks 0 to 5),

1. This chapter describes the enhanced flash memory interface implemented on production parts in the ST40 family. The abbreviation for the enhanced flash memory interface was previously FMI.

- boot from MPX,
- clock master and slave support for MPX (host and satellite support for MPX),
- peripheral support (SRAM and ROM) (connectable banks 0 to 5),
- external bus multi-master (DMA, MPX and other) support,
- slave mode statically set by a device pin,
- power-down support: EMI is able to issue a self-refresh command to SDRAMs before a power-down,
- transparent support for both big and little endianness system.

The EMI memory map is divided into six regions (EMI banks) which may be independently configured to accommodate one of SRAM, ROM, burst flash, SDRAM or an MPX device.

Each bank can only accommodate one type of device, but different device types can be placed in different banks to provide glue-less support for mixed memory systems.

EMI endianness is statically set with the rest of the system following reset and cannot be changed dynamically. Bit positions are numbered left to right from the most significant to the least significant. Thus in a 32-bit long word the left-most bit, bit 31, is the most significant bit and the right-most bit, bit 0, is the least significant.

A maximum of two banks may be configured as SDRAM at the same time, though these banks may be address subdecoded to provide glueless connection to several devices.

If only one bank is dedicated to SDRAM, then subdecoding up to four subbanks is possible. If two SDRAM banks are used, each bank will allow subdecoding for up to two subbanks.

The external data bus can be configured so each bank can be either 32, 16 or 8 bits wide.

Note: *An EMI bank can be address subdecoded into two subbanks. Each subbank has the configuration properties of the bank that is subdecoded.*

Another bank's chip select strobe is used as the subbank chip select, so that there may be two chip selects per bank, one for each subbank. This allows up to four SDRAM devices to be address-mapped into the address spaces of two EMI banks and two peripherals to be mapped into the address space of another bank.

2.3 Address map

The EMI is allocated a region of 128 Mbytes of the available memory map. This is separated into two spaces, one being configuration space used to control the behavior of the EMI, the other a memory region which is mapped onto six user configurable memory banks.

Following reset the EMI is organized as shown in *Table 18*.

Register name	Description	Type	Address offset (hex bytes)	Access size
BANK_0_BASE_ADDRESS	Accesses to this address space cause transfers on EMI bank 0	RW	0x0000 0000 to 0x00FF FFFF	-
BANK_1_BASE_ADDRESS	Accesses to this address space cause transfers on EMI bank 1	RW	0x0100 0000 to 0x01FF FFFF	-
BANK_2_BASE_ADDRESS	Accesses to this address space cause transfers on EMI bank 2	RW	0x0200 0000 to 0x02FF FFFF	-
BANK_3_BASE_ADDRESS	Accesses to this address space cause transfers on EMI bank 3	RW	0x0300 0000 to 0x03FF FFFF	-
BANK_4_BASE_ADDRESS	Accesses to this address space cause transfers on EMI bank 4	RW	0x0400 0000 - 0x04FF FFFF	-
BANK_5_BASE_ADDRESS	Accesses to this address space cause transfers on EMI bank 5	RW	0x0500 0000 - 0x07EF FFFF	-
Unused			0x0600 0000 - 0x07EF FFFF	-
Configuration	Information which defines the operation of each bank, see *Table 19 on page 54*	-	0x07F0 0000 - 0x07FF FFFF	-

Table 18: EMI reset organization

The configuration address space is organized as shown in *Table 19*.

Register name	Description	Type	Address offset	Access size
Reserved (EMI.VCR)	Reserved for version control EMI register	RO	0x00	64
EMI.STATUSCFG	Status register (config flags update), see *Table 38 on page 123*	RO	0x10	32
EMI.STATUSLOCK	Lock register (config flags lock), see *Table 39 on page 124*	RO	0x18	32
EMI.LOCK	Lock register, see *Table 40 on page 125*	RW	0x20	32
EMI.GENCFG	General purpose configuration register set, see *Table 41 on page 126*	RW	0x28	32
EMI.SDRAMNOPGEN	Generate NOP commands during the initialization phase of SDRAM until SDRAMINIT is set, see *Table 42 on page 127*	WO	0x30	32
EMI.SDRAMODEREG	SDRAM configuration data, see *Table 43 on page 128*	WO	0x38	32
EMI.SDRAMINIT	Initialize for SDRAM, see *Table 44 on page 129*	WO	0x40	32
EMI.REFRESHINT	Refresh interval for SDRAM, see *Table 45 on page 130*	WO	0x48	32

Table 19: EMI configuration registers

Register name	Description	Type	Address offset	Access size
EMI.FLASHCLKSEL	Select clock speed for flash devices, see *Table 46 on page 131*	WO	0x50	32
EMI.SDRAMCLKSEL	Select clock speed for SDRAM devices, see *Table 47 on page 132*	WO	0x58	32
EMI.MPXCLKSEL	Select clock speed for MPX devices, see *Table 48 on page 133*	WO	0x60	32
EMI.CLKENABLE	Enable clock generation for different devices, see *Table 49 on page 134*	WO	0x68	32
Reserved		-	0x70 to 0xF8	-
EMI.BANK0	Bank0 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	RW	0x100 to 0x138	-
EMI.BANK1	Bank1 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	RW	0x140 to 0x178	-
EMI.BANK2	Bank2 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	RW	0x180 to 0x1B8	-
EMI.BANK3	Bank3 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	RW	0x1C0 to 0x1F8	-

Table 19: EMI configuration registers

Register name	Description	Type	Address offset	Access size
EMI.BANK4	Bank4 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	RW	0x200 to 0x238	
EMI.BANK5	Bank5 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	RW	0x240 to 0x278	
Reserved			0x280 to 0xFFF8	

Table 19: EMI configuration registers

Each EMI.BANK[0:5] contains a set of 32-bit EMICONFIGDATA registers that are used to configure each bank depending on the type of device that is connected.

The EMI supports three memory types:

- non-multiplexed address and data bus (SDRAM/peripheral/flash interface)
- MPX interface,
- SDRAM interface.

Each memory type has own set of EMICONFIGDATA register formats. *Table 20* describes how the configuration region of each bank is divided.

Register format name	Memory type			Address offset	Access size
	Peripheral	MPX	SDRAM		
EMICONFIGDATA0	See *Table 50 on page 135*	See *Table 54 on page 143*	See *Table 58 on page 146*	0x00	32
EMICONFIGDATA1	See *Table 51 on page 138*	See *Table 55 on page 144*	See *Table 59 on page 148*	0x08	32
EMICONFIGDATA2	See *Table 52 on page 139*	See *Table 56 on page 145*	See *Table 60 on page 148*	0x10	32
EMICONFIGDATA3	See *Table 53 on page 141*	See *Table 57 on page 145*	See *Table 61 on page 148*	0x18	32
Reserved				0x20 to 0x38	

Table 20: EMI.BANK[0:5] registers

The type and organization of each set of bank registers is dependent on the value in EMI.CONFIGDATA0.DEVICETYPE which defines the type of memory or device attached to that bank. The memory types and their associated control registers are described later in this chapter.

2.4 Operation

The EMI is a highly flexible memory device which supports a large range of memory components seamlessly. It accepts memory operations from the system and, depending on the address of the operation, either accesses internal configuration space or one of the possible six external memory banks.

The position, size, clock frequency and memory type supported in each bank is dependent on how the associated control registers, EMI.BANK[0:5], are programmed.

Following reset, all banks start with the same configuration which has allows the system to boot from a large range of non-volatile (or MPX) memory devices.

As part of the boot process, the user should program the EMI configuration registers to match the memory supported in that system, defining the memory size, the location in the address and the device type connected.

2.4.1 Supported transactions

The EMI accepts the following operations from the system when accessing memory devices:

- load 1/24/8/16/32 bytes (LD1, LD2, LD4, LD8, LD16, LD32),
- store 1/2/4/8/16/32 bytes (ST1, ST2, ST4, ST8, ST16, ST32),
- READMODWRITE 4/8 bytes (RMW4, RMW8),
- swap 4/8 bytes (SWP4/8),
- load group 1/24/8/16/32 bytes (LDG1, LDG2, LDG4, LDG8, LDG16, LDG32),
- store group 1/2/4/8/16/32 bytes (STG1, STG2, STG4, STG8, STG16, STG32).

The READMODWRITE transaction comprises three phases:

1. The system passes a READMODWRITE request with an address whose location the initiator wants to read.
2. The EMI processes the request and provides the readdata to the system. Afterwards the EMI does not accepts any further request until the second part of the READMODWRITE request arrives. EMI is locked.
3. The second half of the READMODWRITE request with some data on the wrdata bus is presented. Then the EMI writes the previous location with the new data and signals the completion of the transaction. EMI is now available for new operations (or to leave the mastership of the buses to the external DMA).

The swap transaction comprises two phases:

1 The system passes a swap request with an address whose location the initiator wants to swap, and data on the writedata bus (for the swap).

2 The EMI reads the data stored at the required location, put these data on the readdata bus and writes the wrdata bus content in it, realizing the required swap.

During the processing of READMODWRITE or swap transactions, if the EMI is a bus master, it will not grant the access to any DMA request until the operations are successfully completed. Besides, if EMI is a slave it will not release the control of the bus until the operations are successfully concluded. This will guarantee the atomic nature of these transactions.

For accesses to the configuration space, it supports the following operations:

- load 4 bytes (LD4),
- store 4 bytes (ST4).

All other operations will cause an error response to be generated.

2.4.2 Data organization

The EMI endianness model is statically determined following reset and supports both little and big endian system models. The EMI supports external memory widths of 8-, 16- and 32-bits for most memory types with the exception of MPX where only 32-bit interfaces are supported.

Examples of how the data for various memory transactions is mapped onto memories of 8-, 16- or 32-bits width are shown below:

For subword transfers the following tables show how data alignment and data length conversion is performed depending on the endianness that has been set.

32-bit memory device

Datum	Byte lane	Little endian				Big endian			
		3	2	1	0	3	2	1	0
4 byte, address + 0		MSB	MID2	MID1	LSB	MSB	MID2	MID1	LSB
2 byte address + 0				MSB	LSB	MSB	LSB		
2 byte address + 2		MSB	LSB					MSB	LSB
1 byte[A], address + 0					LSB	MSB			
1 byte, address + 1				LSB			MSB		
1 byte, address + 2			LSB					MSB	
1 byte, address + 3		LSB							MSB

Table 21: 32-bit data width for big and little endian operations

A. For a single byte quantity the MSB and LSB are equivalent.

16-bit memory device

Datum	Byte lane	No of operations	Little endian		Big endian	
			1	0	1	0
4 byte, address + 0 **4 byte, address + 2**		1	MID1	LSB	MSB	MID2
		2	MSB	MID2	MID1	LSB
2 byte address + 0		1	MSB	LSB	MSB	LSB
1 byte[A], address + 0		1		LSB	LSB	
1 byte, address + 1		1	LSB			LSB

Table 22: 16-bit data width for big and little endian operations

A. For a single byte quantity the MSB and LSB are equivalent.

8-bit memory device

Datum	Byte lane	No of transfers	Little endian	Big endian
4 byte, address + 0		1	LSB	MSB
4 byte, address + 1		2	MID1	MID2
4 byte, address + 2		3	MID2	MID1
4 byte, address + 3		4	MSB	LSB
2 byte address + 0		1	LSB	MSB
2 byte, address + 1		2	MSB	LSB
1 byte[A], address + 0		1	LSB	LSB

Table 23: 8-bit data width for big and little endian operations

A. For a single byte quantity the MSB and LSB are equivalent.

2.4.3 Clock reconfiguration for synchronous interfaces

Following reset, the clocks for synchronous interfaces are disabled. This is due to the default reset assuming a memory which may be accessed asynchronously.

To access the synchronous memory, the user will set up the configuration state associated with that bank. The user will then program the required clock ratio in the register EMI.XXXCLKSEL associated with that memory type.

The external clocks, and associated clock dividers, are then enabled by writing 1 to the register EMI.CLOCKENABLE. Once enabled, any attempt to reprogram the clock ratios may have undefined effects.

2.4.4 Master/slave mode

The EMI may operate either as an external bus master or an external bus slave. This is determined by the status of the internal signal EMI_SLAVE. EMI_SLAVE is set in the SYSCONF.SYS_CONF2 register. See the product datasheet for details.

Slave mode: EMI_SLAVE = high

In this mode, another device is the bus master and EMI act as a slave.

When the EMI does not have control of the external memory bus, the bus must be claimed from the external master. This is achieved by taking the EMI_HOLD_REQ signal high. EMI_HOLD_REQ is synchronous to the EMI clock, hence the bus master (or the padlogic on its behalf) will need to synchronize this signal to the master internal clock. When EMI_HOLD_ACK is sampled high, the bus master has relinquished the bus and the EMI external access may safely proceed. EMI_HOLD_ACK is on the external device clock and so is synchronized to the EMI clock before it proceeds into the generic EMI block.

There are no pins provided from the external bus master to indicate to the EMI that the bus is required. Therefore, the EMI must relinquish the bus back to the master after either each single or burst word external access has completed and must re-request the bus for each subsequent access.

The EMI_HOLD_REQ and EMI_HOLD_ACK signals in *Figure 3* are the values seen at the EMI SuperHyway interface.

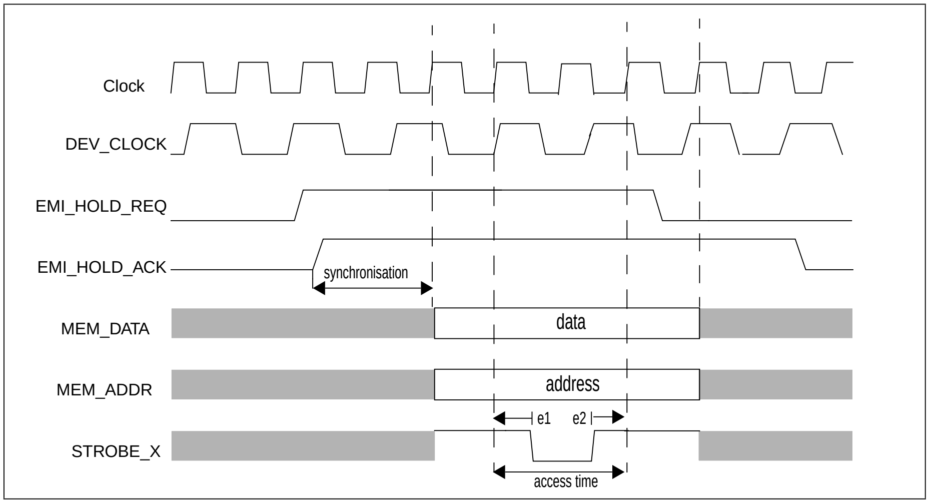


Figure 3: Signal timings

Note: In slave mode the EMI is not supposed either to initialize SFlash™ or SDRAM devices in the system or to refresh any SDRAM device. Therefore the EMI.INITIALIZE register should never be written.

EMI_HOLD_REQ is taken low during the last cycle of the EMI access. So EMI_HOLD_ACK must go low (signalling that the master is driving the bus) at least one cycle after EMI_HOLD_REQ is sampled low, otherwise there could be a bus contention during the last cycle of the access.

Normal mode: EMI_SLAVE = low

In this mode, the EMI is the bus master. The bus arbitration signals EMI_HOLD_REQ and EMI_ HOLD_ACK are not active, EMI_ HOLD_ACK should be held low and EMI_HOLD_REQ will be held low by the EMI block.

A bus request from an external device (from a companion chip (MPX), DMA or any other device) can occur asserting the signal EMI_BUS_REQ**.** However, this input is masked with a configuration bit and so will be ignored unless programmed by software. The purpose of this is to allow an external bus master such as the STi5514 to boot without interruption before another agent can access the bus.

Once the EMI_BUS_REQ HAS BEEN ENABLED, AND BEFORE an external bus request is granted, the generic EMI block will ensure that the current precharge time for the previous SDRAM bank (if present) access is satisfied. The generic EMI block then checks if the bus release time is satisfied for the previous access. The bus release time is the time required for an external device to tri-state the data bus. During this period the DRIVE_ADDRESS and DRIVE_STROBES signals will be set to low. If the bus release time has been satisfied, then the request is granted by taking EMI_BUS_GRANT high.

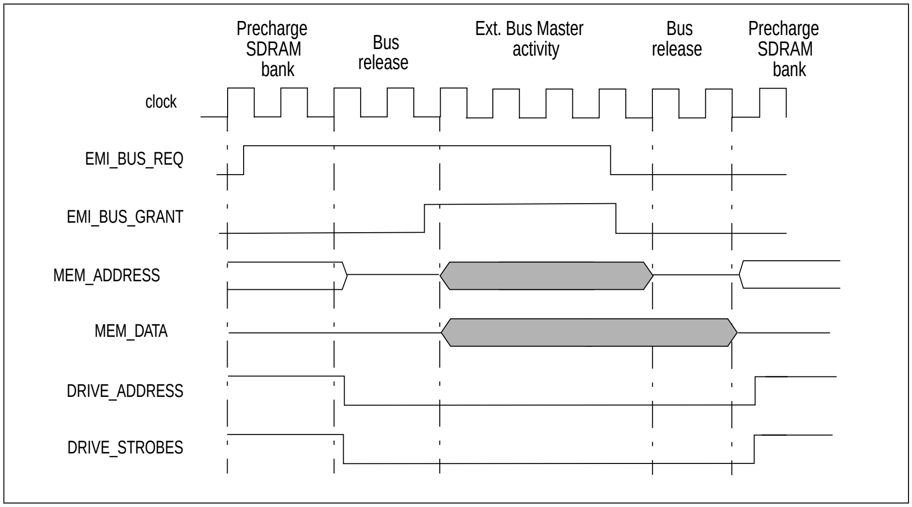


Figure 4: Signal timings

External memory access is terminated by taking EMI_BUS_REQ low. The external agent uses the bus release time (which is fixed) to tri-state the address and data bus before the EMI drives it again. Since the EMI has no idea of the external activity, the slowest memory device is applied before allowing further external accesses. If the bus release time is satisfied the EMI starts to drive the bus again taking the DRIVE_ADDRESS and DRIVE_STROBES signals high. If there is a SDRAM bank, it is then assumed that precharging is required before being accessed (in the case of two SDRAM banks the EMI will choose the highest precharge time to satisfy).

Whilst in the release bus state, the EMI may signal its intent to use the external buses by taking the EMI_RFSHPEND signal high. The external bus is then relinquished by the external agent taking EMI_BUS_REQ low. The external agent must ensure that EMI_BUS_REQ is not taken high again before either the precharge time or largest busrelease time has elapsed. If EMI_BUS_REQ is raised too early, pending EMI accesses will not be given the chance to execute. This is because the request of external device is treated with the higher priority.

2.5 Default and reset configuration

Following reset, a default configuration setting is loaded into all six banks. This allows the EMI to access data from a slow ROM memory or the MPX bus (for MPX boot).

2.5.1 Default configuration for asynchronous boot (EMI_MPX_BOOT = 0)

The default configuration setting is loaded into all six banks on reset. It should allow the EMI to read data from a slow asynchronous ROM (flash) memory. The asynchronous boot uses the configuration register formats for peripherals, see *Section : Peripheral format on page 135*.

Register	Parameter	Default value
EMICONFIGDATA0	WAITPOLARITY	Active high
	LATCHPOINT	End of access cycle
	DATADRIVEDELAY	10 phases
	BUSRELEASETIME	4 cycles
	CSACTIVE	Active during read only
	OEACTIVE	Active during read only
	BEACTIVE	Inactive
	PORTSIZE	Value of the signal EMI_PRTSZ_INIT
	DEVICETYPE	Peripheral

Table 24: Default configuration

Register	Parameter	Default value
EMICONFIGDATA1	CYCLENOTPHASEREAD	Phase
	ACCESSTIMEREAD	(18 + 2 = 20 cycles)
	CSE1TIMEREAD	0 phases
	CSE2TIMEREAD	0 phases
	OEE1TIMEREAD	0 phases
	OEE2TIMEREAD	0 phases
	BEE1TIMEREAD	3 phases
	BEE2TIMEREAD	3 phases

Table 24: Default configuration

The remaining configuration parameters are not relevant for an asynchronous boot, that is the aim of the default configuration.

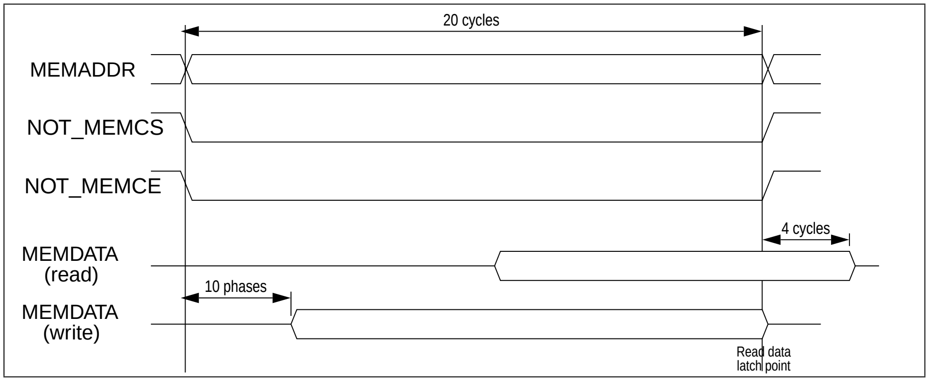


Figure 5: Default asynchronous configuration

2.5.2 Default configuration for MPX boot (EMI_MPX_BOOT = 1)

The default configuration setting is loaded into all six banks on reset. An MPX boot uses the configuration register format for MPX, see *Section : MPX format on page 143*.

Register	Parameter	Default value
EMICONFIGDATA0	BUSRELEASETIME	3 cycles
	DEVICETYPE	MPX
	WAITSTATESREAD	3 cycles
	WAITSTATESWRITE	3 cycles
	WAITSTATESFRAME	One cycle
	EXTENDEDMPX	0 (Hitachi compatible transfer size set)
	WAITPOLARITY	0 (active high)
	STROBESONFALLING	0 (strobes changing on rising edge)
MPXCLKSEL	FLASHCLOCKSELECT	00 (full clock speed)

Table 25: Default configuration for MPX boot

This allows the EMI to read data from a slow MPX (Hitachi compatible) device.

2.6 Peripheral interface with synchronous flash memory support

2.6.1 Overview

A generic peripheral (for example, SRAM, EPROM, SFlash) access is provided which is suitable for direct interfacing to a wide variety of SRAM, ROM, flash, SFlash and other peripheral devices. No subdecoding is possible with banks configured to hold a peripheral configuration.

Figure 6 shows a generic access cycle and the allowable values for each timing field are shown. The strobe timing values for peripherals are shown in *Table 26*.

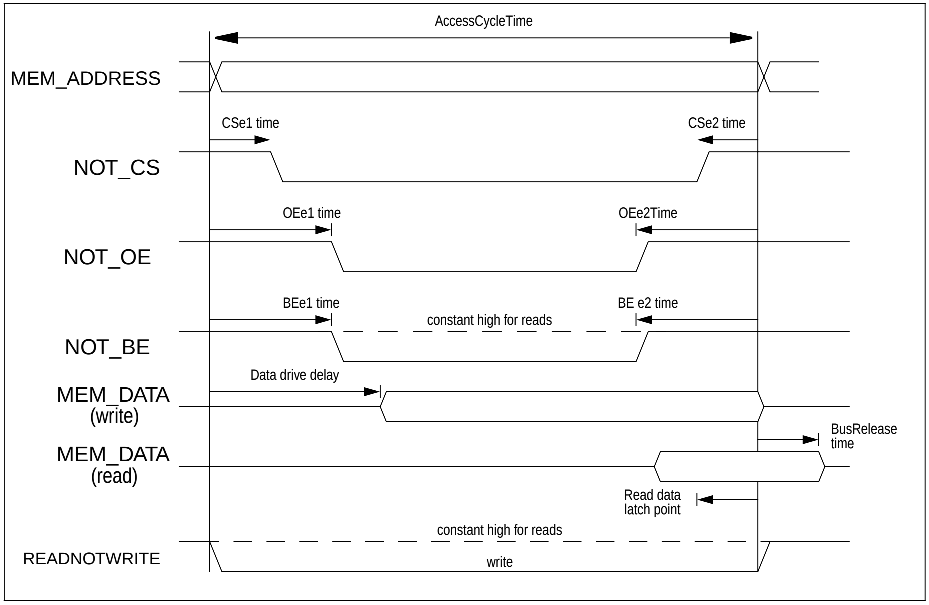


Figure 6: Generic access cycle

Register	Parameter	Programmable value
EMICONFIGDATA0	LATCHPOINT	0: End of access cycle. 1 to 16: 1 to 16 cycles before end of access cycle.
	DATADRIVEDELAY	0 to 31 phases after start of access cycle
	BUSRELEASETIME	0 to 15 cycles
EMICONFIGDATA1 (read parameters) EMICONFIGDATA2 (write parameters)	ACCESSTIMEREAD ACCESSTIMEWRITE	2 cycles + 0 to 125 cycles
	CSE1TIMEREAD CSE1TIMEWRITE	Falling edge of CS 0 to 15 phases or cycles after start of access cycle
	CSE2TIMEREAD CSE2TIMEWRITE	Rising edge of CS 0 to 15 phases or cycles before end of access cycle
	OEE1TIMEREAD CSE2TIMEWRITE	Falling edge of OE 0 to 15 phases or cycles after start of access cycle
	OEE2TIMEREAD CSE2TIMEWRITE	Rising edge of OE 0 to 15 phases or cycles before end of access cycle.
	BEE1TIMEREAD CSE2TIMEWRITE	Falling edge of BE 0 to 15 phases or cycles after start of access cycle
	BEE2TIMEREAD CSE2TIMEWRITE	Rising edge of BE 0 to 15 phases or cycles before end of access cycle

Table 26: Strobe timing parameters for peripherals

Each strobe can be configured to be active on read, writes, neither or both.

CS, OE, BE active code	Strobe activity
00	Inactive
01	Active during read only
10	Active during write only
11	Active during read and write

Table 27: Active code settings

2.6.2 Synchronous burst flash support

Burst mode flash accesses consist of multiple read accesses which must be made in a sequential order. The EMI maps system memory operations onto one or more burst flash accesses depending on the burst size configuration, operation size and the starting address of the memory access.

The EMI supports the following memory devices:

- AMD AM29BL162C,
- ST M58LW064A/B,
- Intel 28F800F3/ 28F160F3.

In *Table 28* there is a brief description and comparison of the main features of the flash memories with which the EMI can interface.

Note: *Not all memory features are supported. When a feature is not supported, this is highlighted.*

	AM29BL162C	STM58LW064A/B	Intel 28F800F3/28F160F3
Size	16 Mbits	64 Mbits	8/16 Mbits
Max[A] operating frequency	40 MHz	60 MHz	60 MHz
Data bus	16 bits fixed	16/32 bits	16 bits fixed
Main operations	Asynch single access write Synch burst read Asynch single access read	Asynch single access write Synch burst read Asynch single access read Asynch page read (not supported by EMI)	Asynch single access write Synch burst read Asynch single access read Asynch page read (not supported by EMI) Synch single access read (not supported by EMI)
Burst size	32 word (not supported by EMI: max burst is 8 words)	1, 2, 4, 8 words[B] or continuous (set by burst config register) (continuous is not supported by EMI)	4 or 8 words or continuous (set by read config register) (continuous is not supported by EMI)
Burst style[C]	Linear burst, 32 words	Sequential burst Interleaved burst (not supported)	Linear burst Intel burst (not supported)
X-latency[D]	70-90-120 ns	7, 8, 9, 10, 12[E] cycles	2, 3, 4, 5, 6 cycles
Y-latency[F]	One cycle	One, two cycles	One cycle
Burst suspend/ resume[G]	Yes using Burst Address Advance (BAA) input	Yes via Burst Address Advance (B) input	No automatic advance
Ready/ busy pin[H]	Yes (RD/BY)	Yes (RD/BY)	No
Ready for burst[I]	No	Yes (R)	Yes (/W)

Table 28: ST, AMD, Intel flash features comparison

A. The flash operating frequency, clock divide ratios and system frequency should be consistent with the maximum operating frequency.

B. A burst length of eight words is not available in the x32 data bus configuration.

C. Modulo burst is equivalent to linear burst and sequential burst. Interleaved burst is equivalent to Intel burst. On AMD the burst is enabled by four asynchronous write operations. On ST and Intel the burst is enabled synchronously using the burst configuration register.

D. X latency is the time elapsed from the beginning of the accesses (address put on the bus) to the first valid data that is output during a burst for ST, while it is the time elapsed from the sample valid of starting address to the data being output from memory for Intel and AMD.

E. 10-12 Only for F = 50MHz

F. Y-latency is the time elapsed from the current valid data that is output to the next data valid in output during a burst

G. In AMD and ST devices, BAA (or B) can be tied active. This means that the address advance during a burst is non-interruptible (Intel likewise). EMI assumes these pins are tied active and does not generate a BAA signal.

H. When the pin is low this means that the device is busy with a program or erase operation. When high, the devise is ready for any read or write operation

I. These signals are used to introduce wait states. For example, in the continuous burst mode the memory may incur an output delay when the starting address is not aligned to a 4 word boundary. In this case a wait is asserted to cope with this delay.

EMI implements a super-set of operational modes so that it is compatible with most of the main functions listed for the three flash families. In the following sections there is a brief description of the EMI flash interface functionality.

2.6.3 Operating mode[1]

Two different programmable read modes are supported:

- asynchronous single read,
- synchronous burst mode (default four words length: configurable to 1-2-4-8 words) using a specific lower frequency clock selected using the EMI.FLASHCLKSEL register.

EMI supports asynchronous single write.

The asynchronous single read/write uses the same protocol as that of the normal peripheral interface.

In *Figure 7* a typical burst access with burst length of four words is shown.

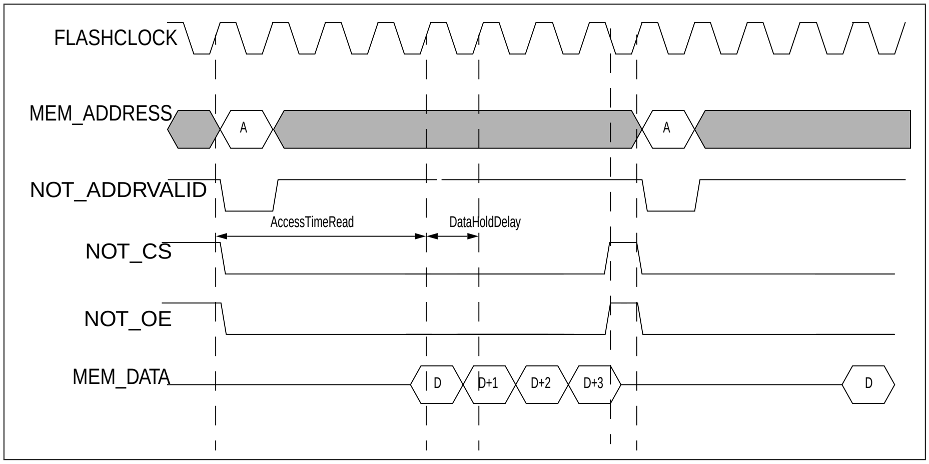


Figure 7: Synchronous burst mode flash read (burst length = 4)

The AccessTimeRead parameter is used to specify the time taken by the device to process the burst request. The rate at which subsequent accesses can be made is then specified by the DATAHOLDDELAY parameter. E1 and E2 delays can also be specified.

2.6.4 Burst interrupt and burst reiteration

The EMI will interrupt the burst after the required amount of data has been read, thus making the chip select of the burst device inactive. This operation is allowed by

1. Continuous burst is not supported by EMI.
   32 words burst size is partially supported by EMI. The burst is interrupted when the required data has been read.
   Asynchronous page mode read is not supported by EMI.
   Interleaved burst mode is not supported by EMI because of the implementation of multiple reads only using synchronous burst mode (feature provided by all the three families of flash chips adopted).

all the three families of flash devices (burst read interrupt for ST device, standby for Intel, terminate current burst read for AMD). Due to this operation, the flash device will put its outputs in tri-state. If a new burst operation is then required, a new chip select and load burst address is provided (EMI_LBA) to the memory chip.

If the flash interface is configured to a burst sequence of [n] bytes, and a burst read request of [i] bytes is presented to the EMI on the ST bus interface, there are three possible outcomes:

- n = i: the EMI performs one burst access during which it gets the exact number of words as requested (see Example A in *Figure 8* with n = i = 8). Depending on the starting address, there will possibly be a wrap that is automatically completed by the flash device. The wrap happens when the starting address is not aligned on a [n]-byte word boundary.

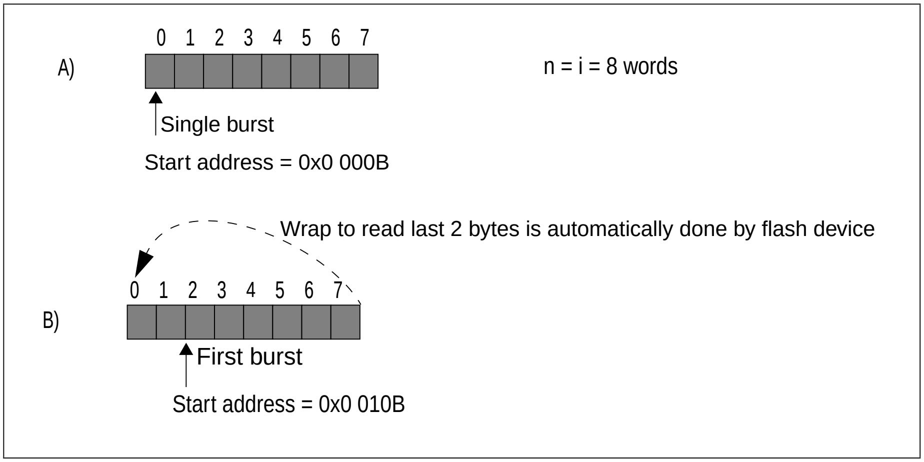


Figure 8: Burst on a flash with a single access

2) n > i: If the starting address is aligned on a [i]-byte word boundary, the EMI will get i bytes from a single burst sequence as explained in the previous paragraph. Then the transfer on flash will be interrupted making inactive the chip select. This will terminate the burst transfer and will put the memory device in stand-by mode, waiting for a new request and starting address for a new burst.

If the starting address is not aligned on a [i]-byte word boundary a first burst on the flash executes until the [i]-byte word boundary is crossed. The burst on the flash is interrupted and there follows another burst with a starting address that wraps to an [i]-byte boundary (directly given by SuperHyway interface) to read the remaining data. After all the required bytes have been read the burst access on flash can be interrupted.

3)n < i: The EMI will need to perform more burst accesses until it gets the required [i] words.

If the starting address is aligned on a [n]-byte word boundary, there are a series of flash burst accesses until the exact number of bytes is met.

If the starting address is not aligned on [n]-byte word boundary, there is a first access on flash to read data until the [n]-byte word boundary is met. This access is then interrupted and new series of accesses are started on a new address provided by SuperHyway (that eventually wraps at the [i]-bytes boundary). This is repeated until the exact number of bytes is reached. This happens in the middle of the last flash burst that will be interrupted in the usual manner.

2.6.5 Synchronous burst enable

This operation is controlled by software and must only be performed when all other configuration registers in the EMI have been programmed.

Table 28: ST, AMD, Intel flash features comparison on page 71, shows that for ST and Intel devices to operate in synchronous burst mode, the configuration parameters must be set in a special configuration register inside the memory device. The configuration software routine starts two asynchronous write operations for each bank of burst memory, where address and data, respect precise configuration rules. However, for AMD the burst enable will be performed by a sequence of four normal asynchronous writes.

2.6.6 Support for lower clock rates

Many SFlash devices operate in the 30 to 50 MHz clock range (*Table 28: ST, AMD, Intel flash features comparison on page 71*) whereas the EMI will operate up to a clock frequency of 100 MHz. To deal with this difference, the EMI needs to run in a lower speed mode. The hardware in the EMI needed for this mode forces accesses to always start on the rising edge of the slower clock. It is up to the user to configure the other EMI timings, to setup and latch, on the appropriate edge of this slower clock.

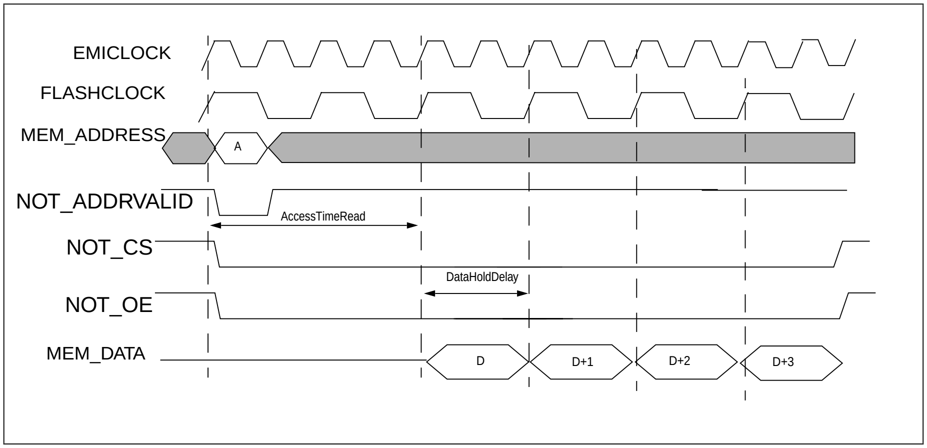


Figure 9: Half speed EMI SFlash clock

2.6.7 Initialization sequence

Peripheral interfaces are used immediately after reset to boot the device. Therefore, the default state must be correct for either synchronous or normal ROM, this is unless the MPX boot is enabled. An SFlash device can be interfaced to normal ROM strobes with the addition of only the ADDRESS VALID signal and the clock. When the CPU has run the initial bootstrap, it can configure both the SFlash device and the EMI to make use of the burst features.

Note: After reset the flash devices are in asynchronous read mode.

Caution:
The process of changing from default configuration to synchronous mode is not interruptible. Therefore the CPU must not be reading from the device at the same time as changing the configuration as there will be a small window where the EMI's configuration will be inconsistent with the memory device's.

2.6.8 Flash subdecoding

As shown in *Table 28: ST, AMD, Intel flash features comparison on page 71*, the maximum size of memory chip for SFlash is 64 Mbits. This may not be enough for some of the variants that use the EMI. Hence subdecoding for flash may be needed. As for the normal peripherals, CHIP SELECT is the signal that starts a transaction.

Therefore, the best choice in term of flexibility, is to leave the address and chip select subdecoding task to the padlogic (product dependent) as usual.

2.6.9 MEM_WAIT

This synchronous pin is sampled on each processor clock cycle during accesses to banks programmed using the peripheral format. In cycles when it is sampled high, the external access is halted and the strobe state does not change. The access resumes when MEM_WAIT is sampled low.

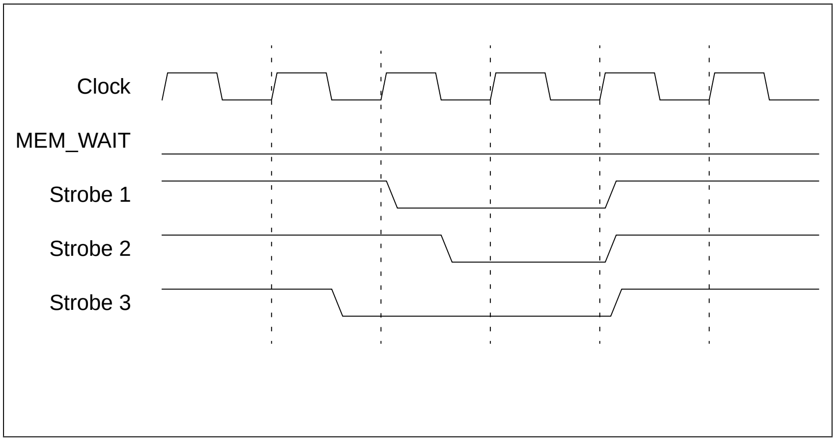


Figure 10: MEM_WAIT sampled high

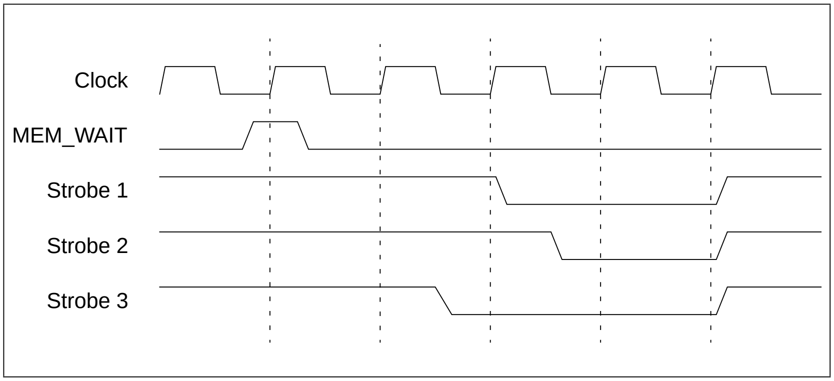


Figure 11: MEM_WAIT sampled low

Note: *MEM_WAIT is ignored if it is sampled high on the last cycle of the peripheral access.*

2.7 MPX interface

MPX is a Hitachi proprietary standard that has been developed as an internal optimized version of the PCI standard. The MPX interface is based on a multiplexed address/data type protocol and enables easy connection with an external companion-chip. Generally, the increased pin count of a companion-chip interface affects system configuration costs. The MPX bus can decrease the pin count maintaining the large bus bandwidth in burst data transfers.

MPX in general allows two companion chips to exchange data in both directions. Both sides can initiate interactions (initiator), but only after gaining the bus mastership. The EMI supports an initiator-only MPX interface with a fixed bus-width of 32 bits.

Two cases are possible:

- EMI is a bus master (statically set at power-on).
  In this case the EMI can access all the six banks as a bus master. One or more of these banks can hold MPX target devices. EMI selects the target MPX bank using the CHIP SELECT signals, while the other signals are shared by all the MPX devices. The EMI can release the bus on demand. If some external master (MPX included) wants to access a memory device, it can ask the EMI to release (tri-state) its outputs on the bus signals and drive them to directly access the memory. For the EMI all the arbitration between external masters is completely transparent, the EMI only deals with the release of the bus when an external request arrives.
- EMI is a bus slave (statically set at power-on).
  The bus master is external. In this case the EMI requires the bus before accessing MPX or any other memory devices.

The EMI cannot be accessed as a target by any external master (including MPX).

All signals on this interface are synchronous to the MPX clock, which can be set to full EMI subsystem clock OR 1/2 OR 1/3 of this clock. EMI.MPXCKSEL sets the MPX clock. In clock slave mode, the MPX clock comes from the MPX clock master.

2.7.1 MPX connection

Figure 12 shows a basic scheme of MPX connections.

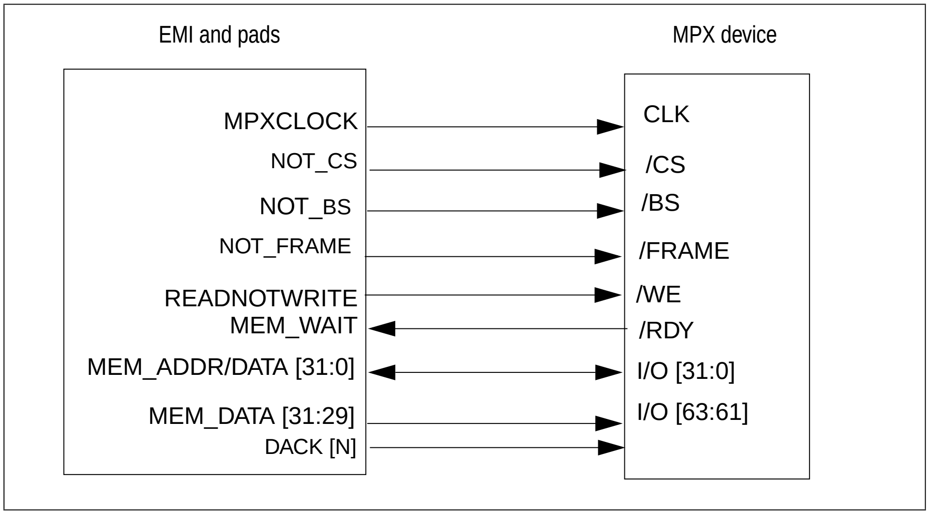


Figure 12: MPX initiator-only interface

MPXCLOCK	Output clock
NOT_CS	Chip select
NOT_BS	Bus start Active at the beginning of the operation and used to latch the access starting address
NOT_FRAME	When active (and no wait inserted) this indicates new data is required. A transfer of [n] words will require NOT_FRAME active at least for [n] cycles
MEM_WAIT (/RDY)	During read/write operations when low data are valid/latched on next cycle, otherwise a wait state is introduced
I/O	I/O bus where data and address are multiplexed.
MEM_DATA [31:29]	Used as command bus to select the size of transfer access. The only supported access sizes are 8-16-32 bits and 32-byte burst

DACK	Used to indicate that transfer is from DMA [n].

The address is output to mem_writedata[28:0] and the access size to MEM_WRITEDATA [31:29] (for simplicity, this bus is referred to on diagrams as MEM_DATA).

Note: *The command information is output on D63-D61 in the SH4 s MPX interface. The* MEM_DATA *[31:29] of the EMI has to be connected with I/O [63:61] of an MPX device. The meaning of these commands bits is explained on Table 29.*

D63	D62	D61	Transfer size
0	0	0	8 bit
0	0	1	16 bits
0	1	0	32 bits
0	1	1	64 bits
1	X	0	16 byte burst[A]
1	X	1	32 byte burst

Table 29: Transfer size on MPX device

A. This and next codings are a EMI dedicated super-set. Actually for Hitachi the 1xx combinations all correspond to 32 byte burst. This super-set can be disabled by configuration.

2.7.2 Endianness in MPX

MPX handles both types of endian. The data is in the lane according to that data address. Byte lane MEM_DATA [31:24] always carries MSB and MEM_DATA [7:0] carries LSB in 32-bit data access. In smaller access such as byte access, byte lane is determined by access address. The byte of address 4n is going to MEM_DATA [31:24] and the byte of address 4n + 3 is going to MEM_DATA [7:0] in big endian case. In little endian case, address 4n goes to MEM_DATA [7:0] and address 4n + 3 goes to MEM_DATA [31:24]. All the EMI has to do is map 64-bit access to two consecutive 32-bit bus accesses (equal to the maximum bus size, in this case 32-bit). The access sequence should be MSBytes first for big endian case and LSBytes first in little endian case.

2.7.3 External and internal wait states insertion

The signal /RDY has been introduced in the interface description. This signal is usually used by a slow MPX device to insert wait states on the other side. External wait has some delay. It is difficult to provide a correct ready value on time, particularly in write operations.To deal with the variable response times of different implementations of external logic, the programmability of internal wait assists. For this reason the EMI MPX interface provides a configurable number of internal wait states for each bank of MPX.

2.7.4 MPX clock

The EMI padlogic provides a clock to all MPX interface modules, unless configured as a clock slave (multimaster mode).

1. EMI releases the external bus to other masters.
2. EMI is configured as a stable slave.

If EMI is configured as a clock master, therefore, other masters must synchronize with the clock which the EMI padlogic provides. At the 100 MHz clock frequency, a companion chip has a PLL inside the design to synchronize with the EMI clock.

When the EMI is configured as a clock slave, the device contains the necessary PLLs to perform clock synchronization.

There follows a list of diagrams explaining the main transactions on the MPX interface.

The shortest read transaction must last at least three clock cycles. The first is to issue the start of operations and latch the address, the second to avoid bus contention (the address comes from the initiator while the data comes from the target), and the third to latch the data arriving from the target. Obviously this access time can be enlarged with external or internal (or both) wait cycles.

The shortest write transaction can last two cycles. The first is to issue the start of operations and latch the address, the second to effectively send the data to the target that will latch them. No dummy cycle is needed between the start and completion of the transaction because both the address and data come from the initiator and there is no bus contention hazard.

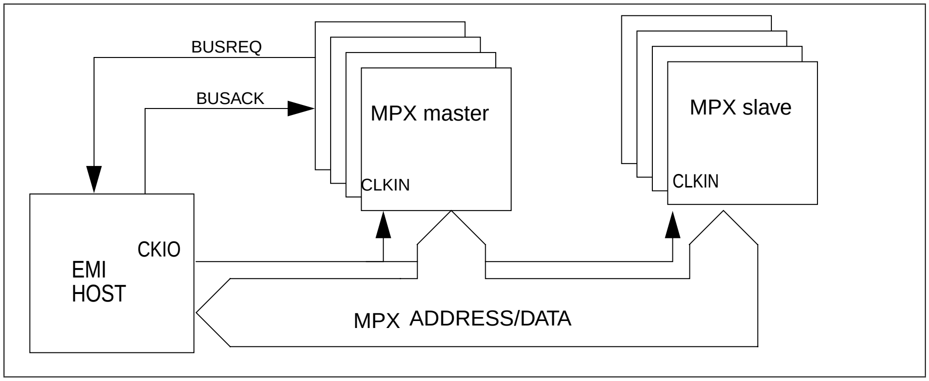


Figure 13: MPX clock connections

2.7.5 MPX interface timings

The diagrams below show a number of example MPX transfers.

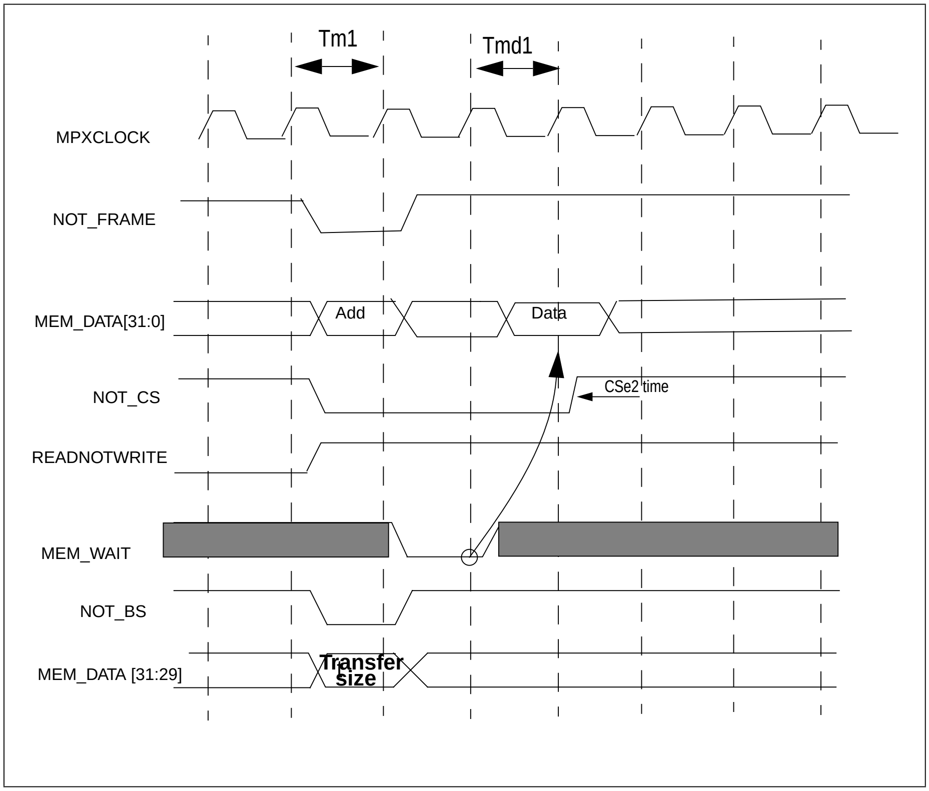


Figure 14: MPX interface timing 1 (single read cycle, no wait)

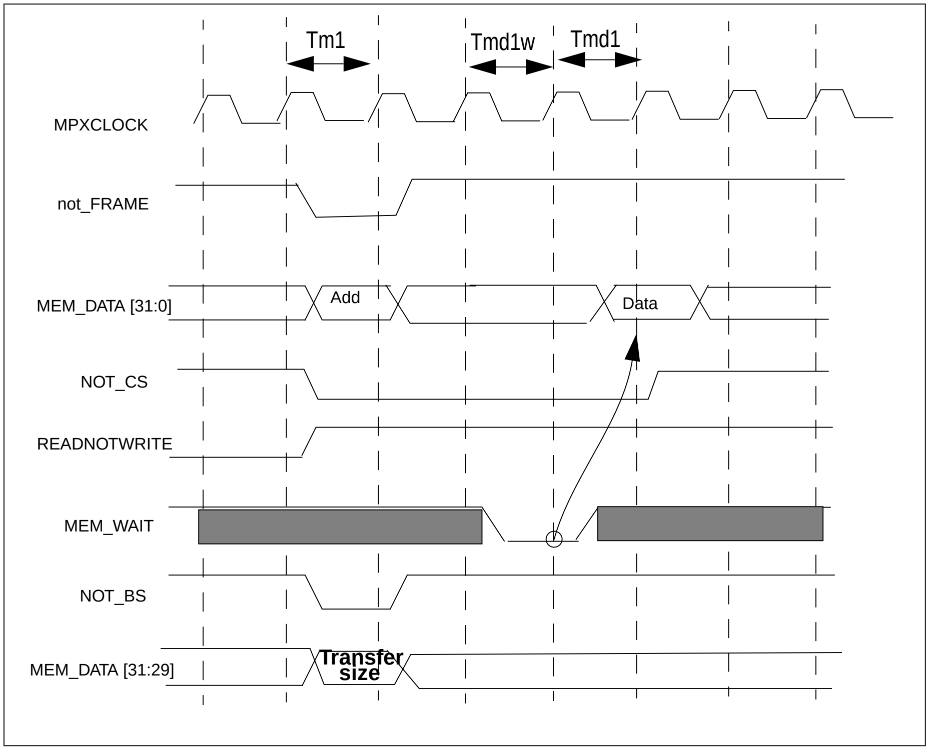


Figure 15: MPX interface timing 2 (single read cycle, one cycle internal wait)

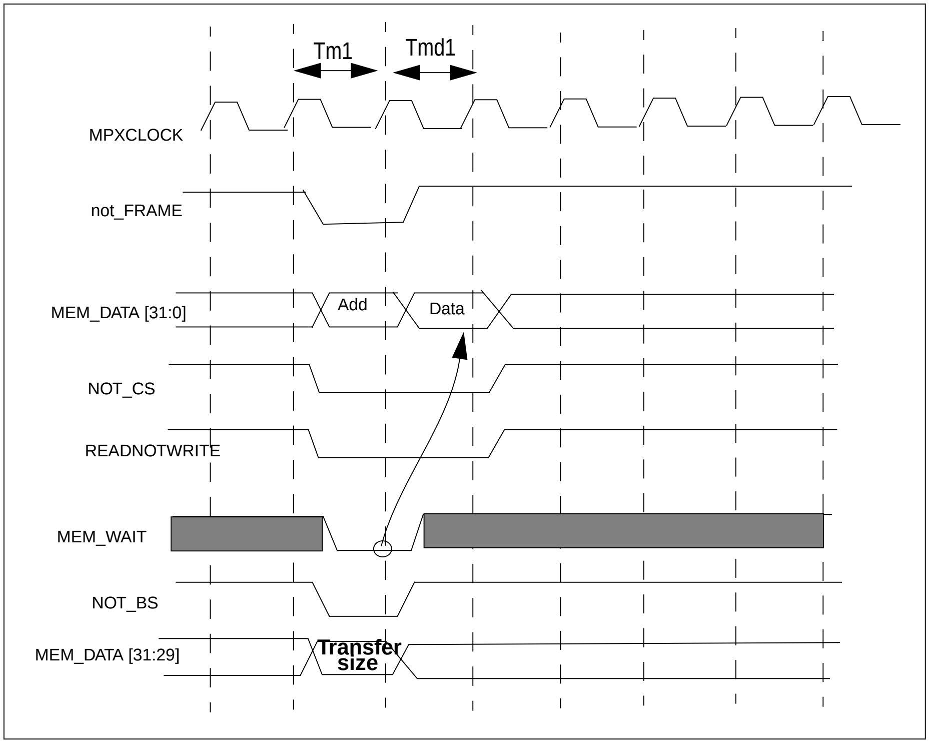


Figure 16: MPX interface timing 3 (single write cycle, no wait cycles)

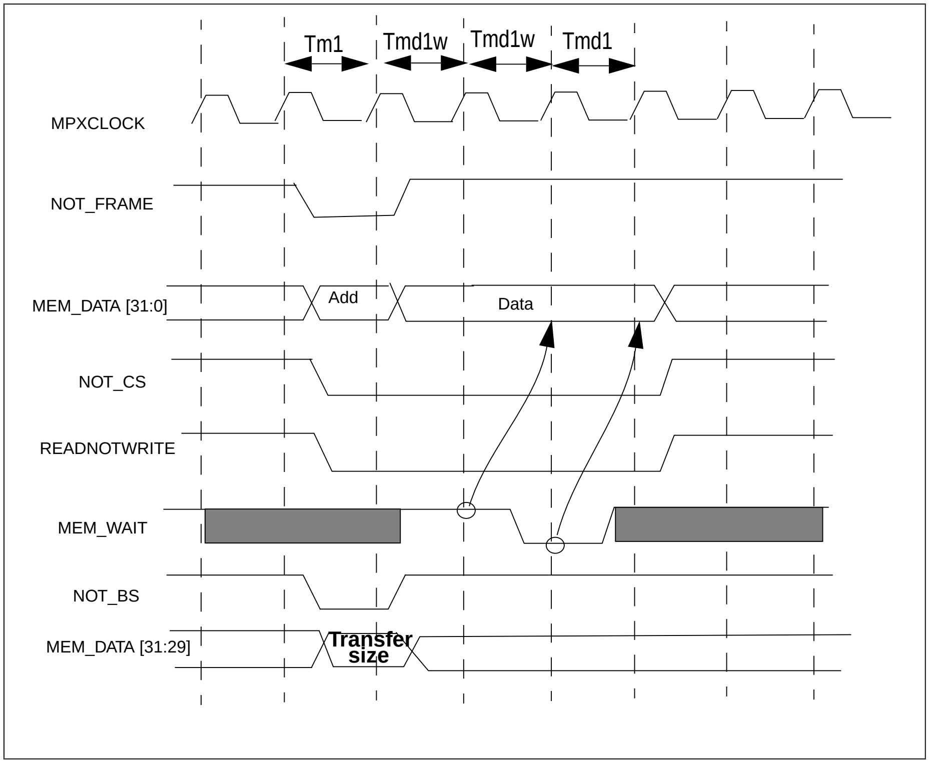


Figure 17: MPX interface timing 4 (single write, one external wait, one internal wait)

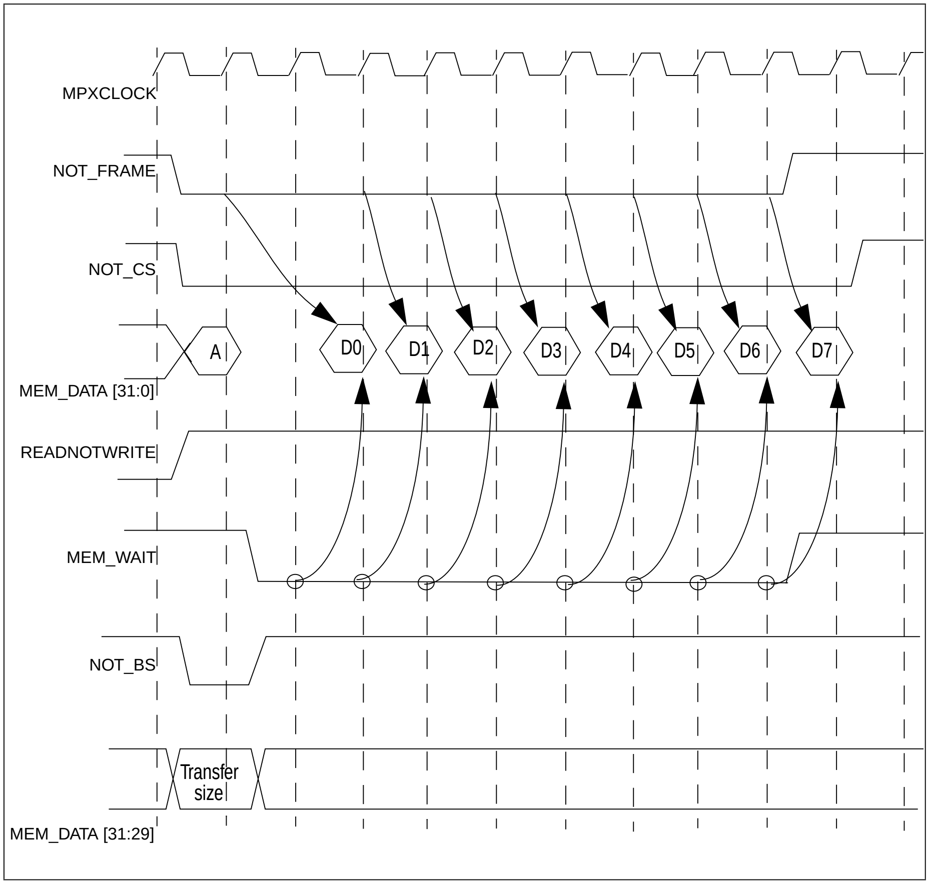


Figure 18: MPX interface timing 5 (burst read cycle, no wait)

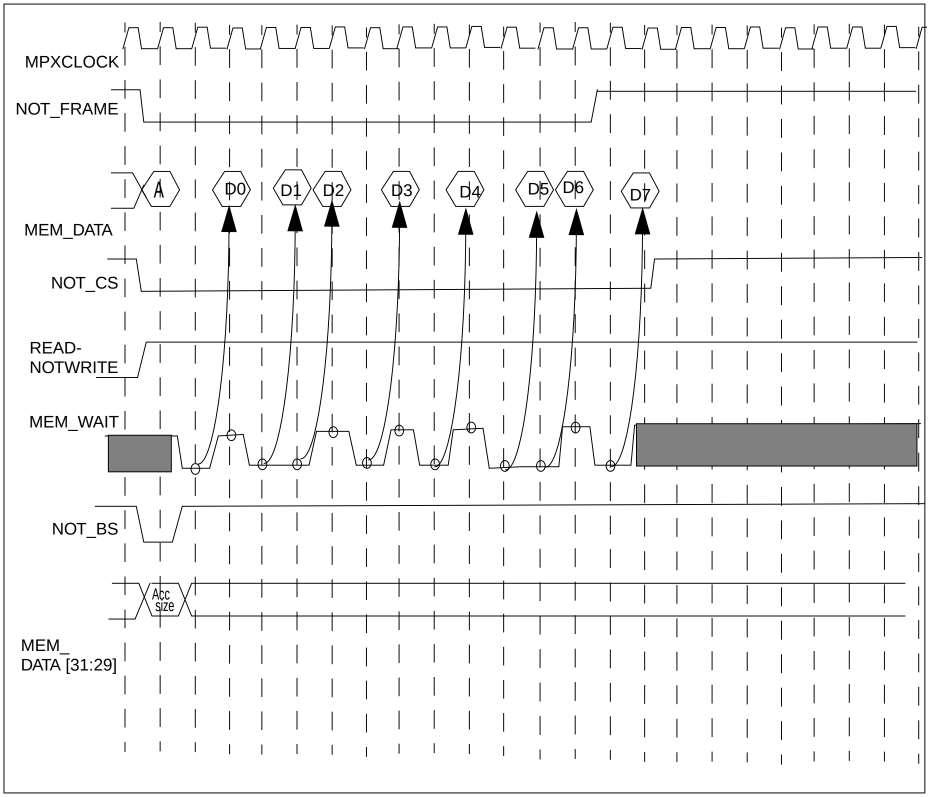


Figure 19: MPX interface timing 6 (burst read cycle, no internal wait, D1, D3, D4, D5, D7 delayed by mem_wait)

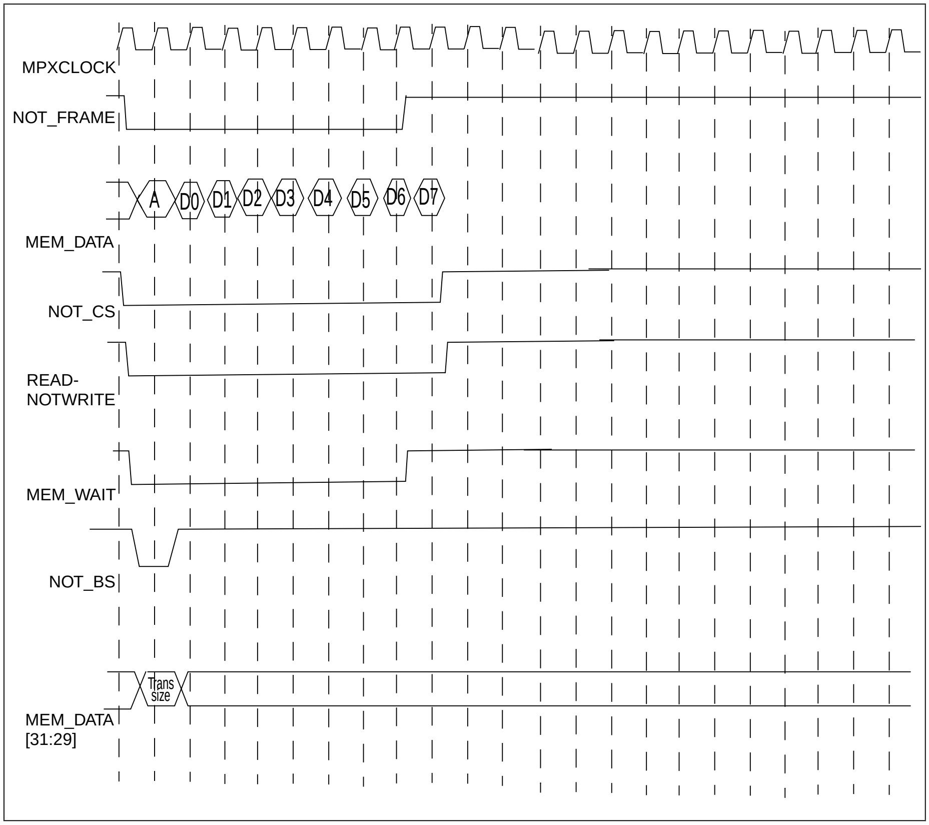


Figure 20: MPX interface timing 7 (burst write cycle, no wait cycle)

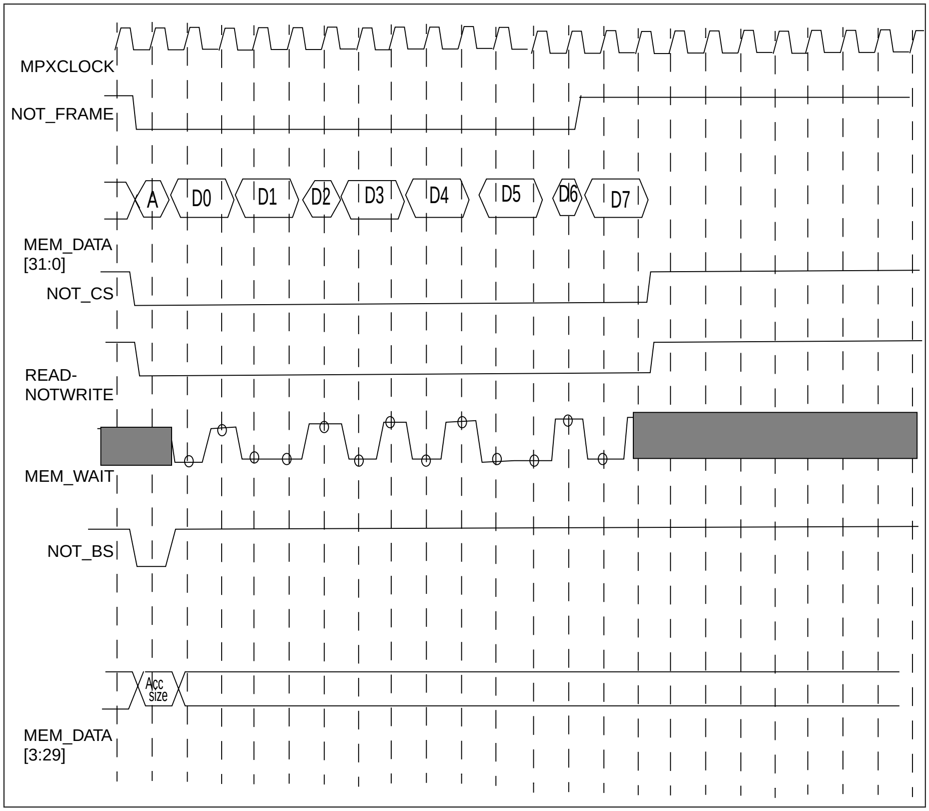


Figure 21: MPX interface timing 8 (burst write cycle, one internal wait for first write and external wait for D1, D3, D4, D5, D7)

Note: */TRDY low means that the data will be accepted (latched) on next cycle. /TRDY high means that the data is not accepted on next cycle and so must stay on the bus for an extra bus cycle.*

2.8 SDRAM interface

All signals on this interface are synchronous to the SDRAM clock, which can be set to the full EMI subsystem clock OR 1/2 OR 1/3 of this clock. The set-up of the EMI.SDRAMCKSEL sets the SDRAM clock.

2.8.1 Typical access

The following diagram describes a typical write access to an SDRAM. The waveforms show what should appear on the pads of a device containing the generic EMI. This example shows a bank activation, due to a page miss, then two write accesses in the same bank which are performed in page mode.

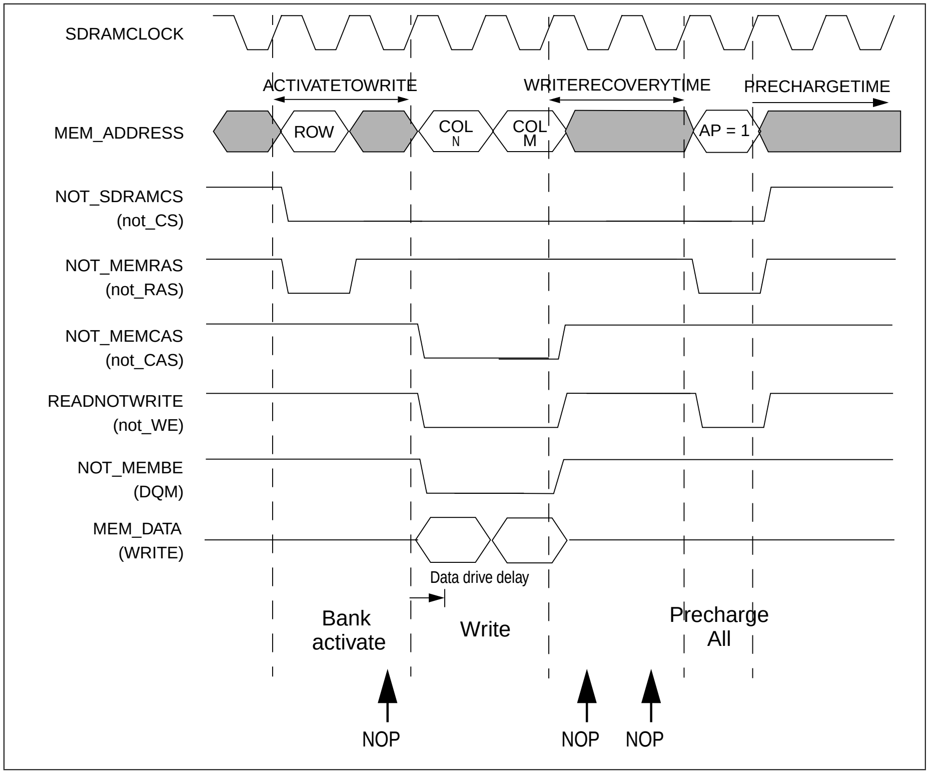


Figure 22: Generic SDRAM write access

A precharge is then completed in anticipation of another bank activation command. If, as in this example, only one SDRAM word is to be written, then the NOT_MEMBE signal is used as a data mask so that only the correct word is updated.

The following figure shows a bank activation, due to a page miss, then two read accesses in the same bank are performed in page mode.

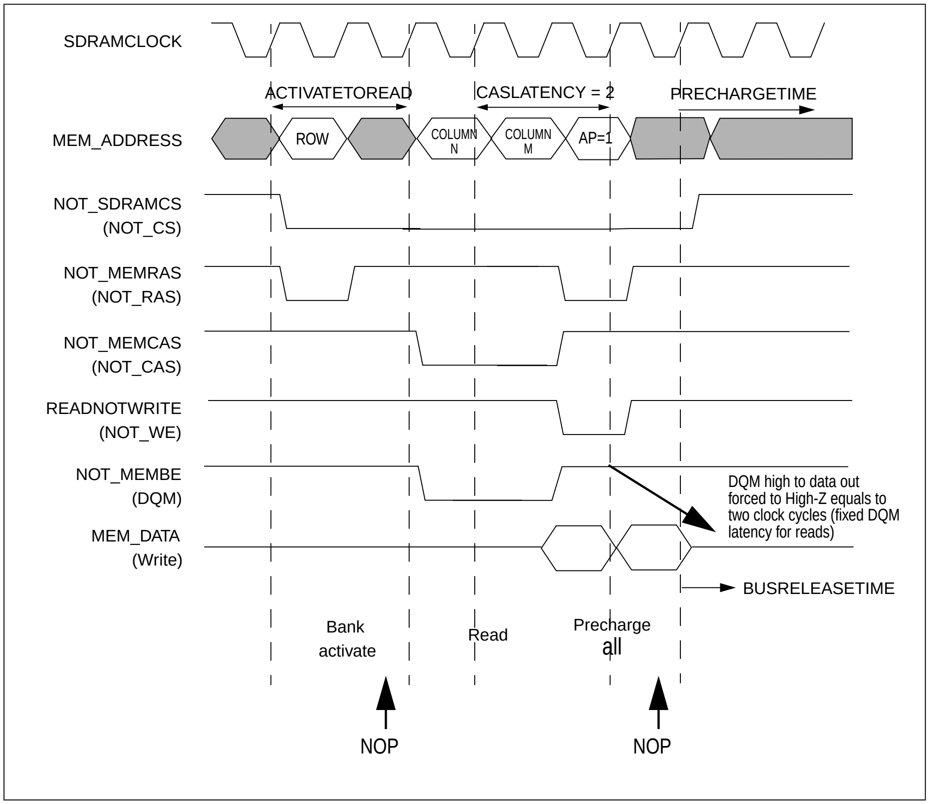


Figure 23: SDRAM read accesses with CASlatency = 2 cycles

A precharge is then completed in anticipation of another bank activation command. If, as in this example, only one SDRAM word is to be read, then the NOT_MEMBE signal is used as a data output enable.

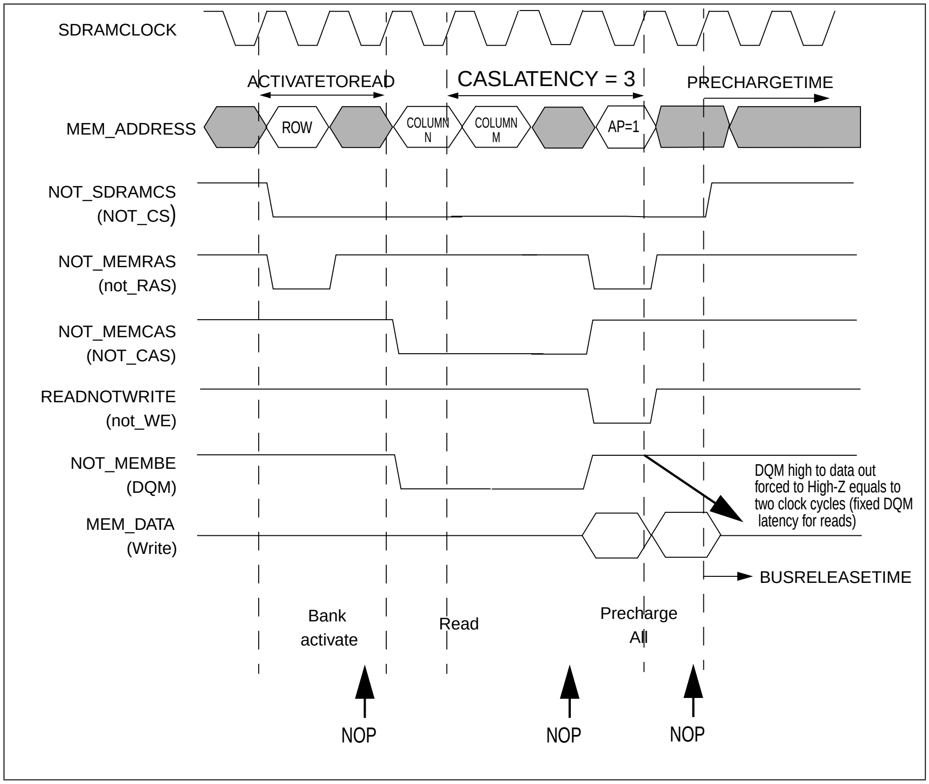


Figure 24: SDRAM read accesses with CASlatency = 3 cycles

2.8.2 Description of signals

MEM_ADDRESS

This is driven with the row address during an activate command and the column address (the full address bus without being shifted) during a read or write operation.

Two address lines have a special meaning to the SDRAM device:

- A10 address line is used as a mode bit during precharge, read and write commands. It is sometimes called the AUTO PRECHARGE (AP) signal.
- The top address line on the memory device is called the BANK SELECT (BS) and must be consistent between opening and accessing the memory page.

As the top address bit changes depending on memory size, all of the address lines above AP could hold their row address throughout read and write commands.

The address lines used for the mode selection are controlled using the EMICONFIGDAT0.PORTSIZE configuration register setting, because the address lines effectively shift as the bus width alters.

MEM_DATA

For writes, data is driven for each cycle of the burst access, either immediately after start of access cycle, or one phase later (EMICONFIGDATA0.DATADRIVEDELAY configuration parameter). For reads, the data will be latched for each cycle of the read. This occurs a number of cycles after the command is sent to the SDRAM (EMICONFIGDATA0.CASLATENCY parameter).

NOT_MEMRAS: NOT_RAS strobe

Normally this is high. However it will be low for a bank activate, precharge, refresh cycle. Additionally it will be low for the SDRAM mode register initialization cycle. NOT_MEMRAS will be shared by all the SDRAM present in the system.

NOT_MEMCAS: NOT_CAS strobe

Normally this is high. However it will be low in a read, write or refresh cycle. Additionally it will be low for the SDRAM mode register initialization cycle. NOT_MEMCAS will be shared by all the SDRAM present in the system.

NOT_SDRAMCS [3:0]: NOT_CS strobes

These signals select which device an access is destined for. They are normally high and 1 is asserted low in the cycle when an access is made.

NOT_MEMBE [3:0]: DQM strobes

Normally this is high. NOT_MEMBE is asserted low during read and write accesses to enable which bytes or words are accessed.

DQM is a multiple function signal defined as data mask for both reads and writes. During reads, DQM performs synchronous output enable. During writes, DQM performs write data masking. The DQM latency is different for reads and writes. For reads, DQM latency is defined as the difference between the clock when DQM is asserted and the clock when the output bus has been forced to High-Z. Its value is always two clock cycles. For writes, DQM latency is defined as the difference between the clock when DQM is asserted and the clock when the write input data is inhibited; its value is always 0.

NOT_MEMBE[N] is only active if the corresponding byte is to be accessed. For single byte accesses only 1 is active. The behavior of these signals is dependent on the CONFIGDATA0.PORTSIZE configuration bits.

READNOTWRITE: NOT_WE strobe

This signal indicates that the current cycle is a read cycle and is normally high. It is asserted low for the precharge and write commands. Additionally it will be low for the SDRAM mode register initialization cycle.

2.8.3 SDRAM controller states

The following diagram shows how the generic EMI controls an SDRAM. These states are a subset of the commands implemented by an SDRAM. The diagram describes the functionality and does not explain the actual implementation, that is these states may not all be implemented by one block within the EMI. For instance the refresh controller will take over from the SDRAM controller when it decides it is time for a refresh operation.

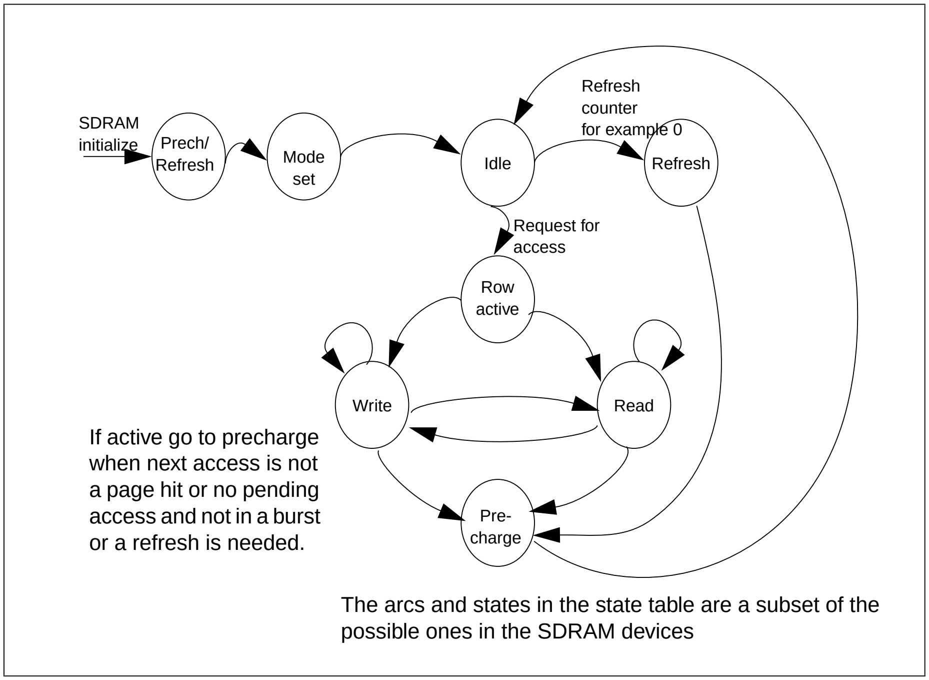

SDRAM initialize
Prech/ Refresh
Mode set
Idle
Refresh counter for example 0
Refresh
Request for access
Row active
Write
Read
If active go to precharge when next access is not a page hit or no pending access and not in a burst or a refresh is needed.
Pre-charge
The arcs and states in the state table are a subset of the possible ones in the SDRAM devices

Figure 25: SDRAM controller states in the current EMI implementation

2.8.4 Supported SDRAM commands

The generic EMI supports the following SDRAM commands. The relevant NOT_SDRAMCS signal will be active for all these operations.

Command	Strobes state			Mode bits		Address	Notes
	NOT_RAS	NOT_CAS	NOT_WE	BS	A10 AP		
PRECHARGE ALL	0	1	0	X	1	X	Used to close pages after a burst in simple page mode
ACTIVATE	0	1	1	1/0	1/0	Row addr	Opens a page (row active)
WRITE	1	0	0	1/0	0	Col. addr	Write words (uses byte enables to mask bytes)
READ	1	0	1	1/0	0	Col. addr	Read words
MODE SET	0	0	0	1/0	1/0	Mode data	Done once only at start up
REFRESH	0	0	1	X	X	X	CBR style refresh (auto refresh) OR self refresh[A]
NO OPERATION	1	1	1	X	X	X	Does nothing (same as having CS inactive)

Table 30: SDRAM commands with NOT_SDRAMCS active

A. If clock enable is high, this is auto-refresh, else this is self-refresh

2.8.5 Supported operations applicable to a single bank of SDRAM

Using the above set of commands, the current implementation of the EMI will support the following truth table in *Table 31*.

SDRAM current state	Command	Notes
Idle	DESL	DESL means NOT_SDRAMCS not active.
	ACTIVATE	The bank specified by the address pins and row address is activated.
	REFRESH	SDRAM enters in refresh mode.
	SELF_REFRESH	The SELF_REFRESH command is issued and EMI goes in power-down.
	MODE SET	SDRAM enters in mode register set cycle.
Row active	NOP	When the number of cycles between the activate of the page and the read/write command is > 1 cycle.
	READ	A read operation starts.
	WRITE	A write operation starts.
Read	READ	This command is issued for each read access in the same row (page mode).
	NOP	This command is performed when a write access in the same row follows the current read access (page mode) or before a PRECHARGE ALL command when EMICONFIGDATA3.CASLATENCY is > 2 clock cycles. If in the cycle in which the NOP is issued a new request comes and it is in the same row, a read or a write command is generated.
	PRECHARGE ALL	This command happens each time there is a new read or write access to a different row address (page miss) or an EMI bank switch or no new pending requests from the EMI buffer or a pending refresh or a DMA access. The command is issued in the cycle after the current read only if EMICONFIGDATA3.CASLATENCY is < 3 clock cycles.

Table 31: Supported commands for a single SDRAM access

SDRAM current state	Command	Notes
Write	WRITE	This command is issued for each write access in the same row (page mode).
	READ	This command happens when a read access follows a write access and is in the same row (page mode).
	NOP	This command happens each time there is either a new read or write access to a different row address (page miss) or an EMI bank switch or no new pending requests from the EMI buffer or a pending refresh or a DMA access. The command will be followed by a PRECHARGE ALL after the WriteRecoveryTime is expired. If in the cycle in which the NOP is issued a new request comes and it is in the same row, a read or a write command is generated.
Precharge All	DESL	DESL means NOT_SDRAMCS not active.
	NOP	This command will be performed until DESL command.
Refresh	DESL	DESL means NOT_SDRAMCS not active.

Table 31: Supported commands for a single SDRAM access

2.8.6 Multiple banks

If subdecoding is used, the generic EMI block can support up to four banks of SDRAMs in the same EMI bank. The NOT_MEMRAS and NOT_MEMCAS strobes are shared by all the devices. Bank selection is done using the NOT_SDRAMCS[3:0] signals.

One SDRAM bank

In this case only one bank is connected to an SDRAM.

Subbanks	NOT_SDRAMCS[n]
1[A]	Only NOT_SDRAMCS0 is used.
2	Subdecode address: 0: NOT_SDRAMCS0 1: NOT_SDRAMCS1
4	Subdecode addresses: 00: NOT_SDRAMCS0 01: NOT_SDRAMCS1 10: NOT_SDRAMCS2 11: NOT_SDRAMCS3

Table 32: NOT_SDRAMCS[n] usage with one subdecoded SDRAM bank

A. The EMI bank is not subdecoded

Two SDRAM banks.

EMI bank	Number of EMI subbanks	NOT_SDRAMCS[n]
0	1[A]	NOT_SDRAMCS0
1	1[A]	NOT_SDRAMCS2
0	2	NOT_SDRAMCS0 NOT_SDRAMCS1
1	1[A]	NOT_SDRAMCS2
0	1[A]	NOT_SDRAMCS0
1	2	NOT_SDRAMCS2 NOT_SDRAMCS3
0	2	NOT_SDRAMCS0 NOT_SDRAMCS1
1	2	NOT_SDRAMCS2 NOT_SDRAMCS3

Table 33: NOT_SDRAMCS[n] usage with two subdecoded SDRAM banks

A. The EMI bank is not address subdecoded

2.8.7 Burst access behavior

A page hit occurs when a new memory access is in the range of an address in a page already activated. The EMI will always access the SDRAM device in page mode, if enabled independently (that is RASBITS not all set to 0), by the value of the burst length specified in the mode register.

The EMI will not implement the JEDEC standard for SDRAM devices about random column accesses, also called the 2n-rule.

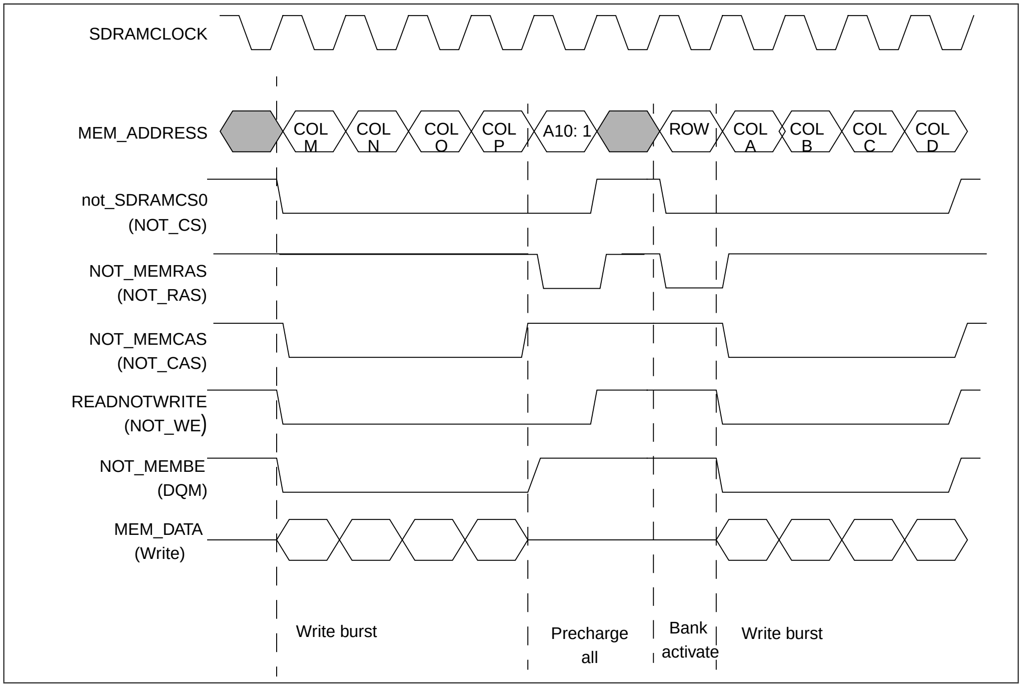


Figure 26: Case of burst write accesses (BL = 2, PORTSIZE = 16 bits)

The activate command is used to open a page and allow read and write operations to be completed.

The EMI will analyze the access for a page hit, and if no following access is available the SDRAMs will be precharged. This will incur a precharge and activate delay before making the next access.

2.8.8 SDRAM accesses example

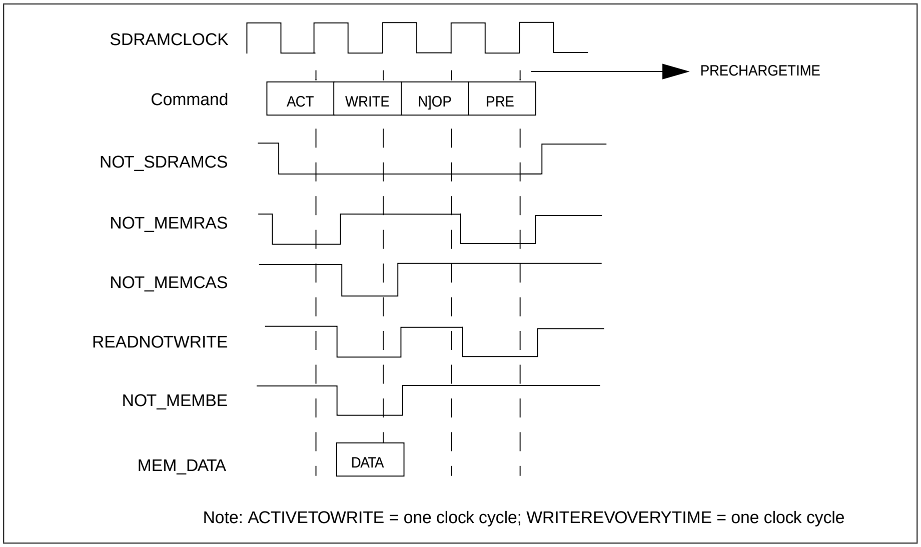


Figure 27: Single WRITE

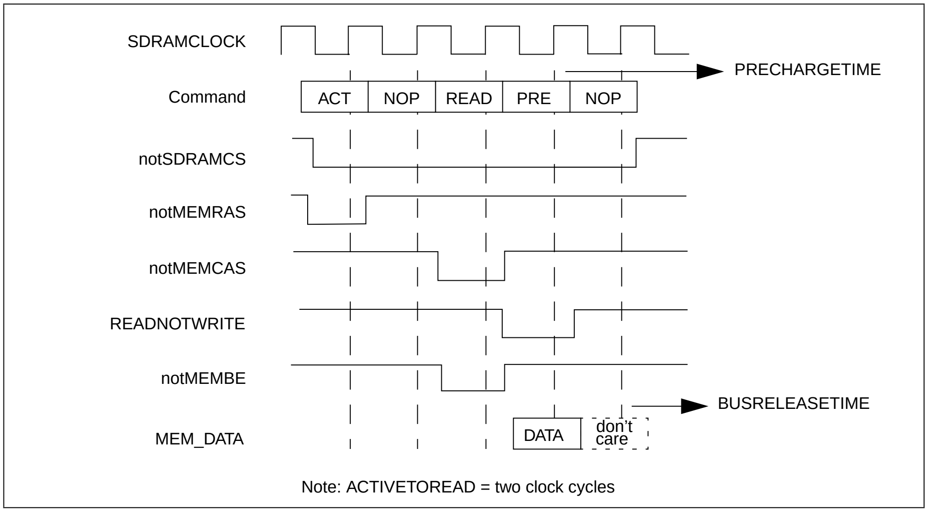


Figure 28: Single read and CAS latency = 1 cycle

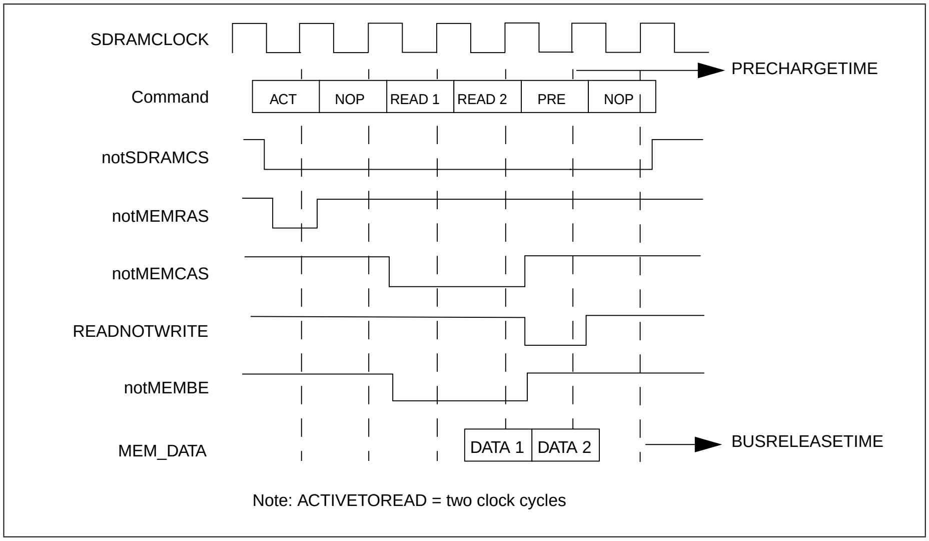


Figure 29: Page mode reads and CAS latency = 1 cycle

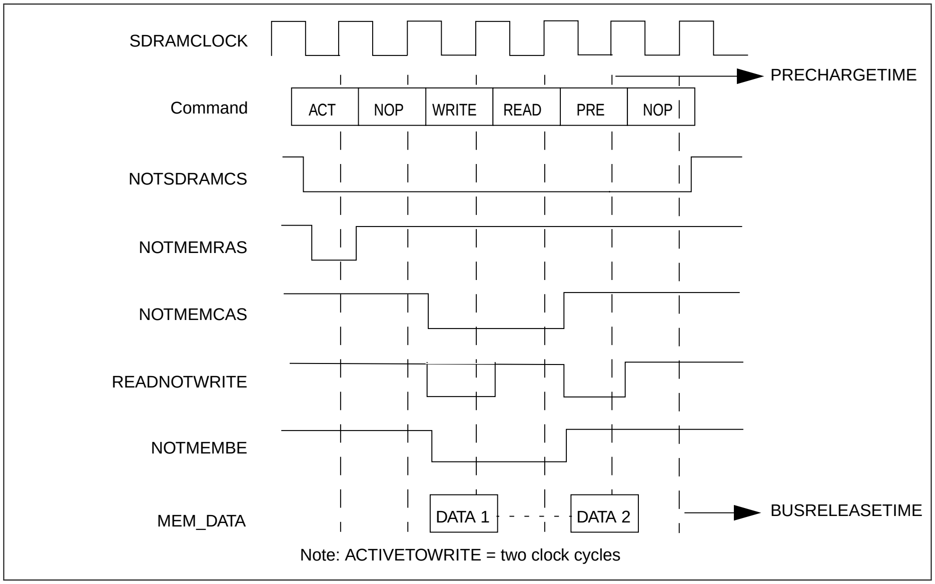


Figure 30: Page mode read after write and CAS latency = 1 cycle

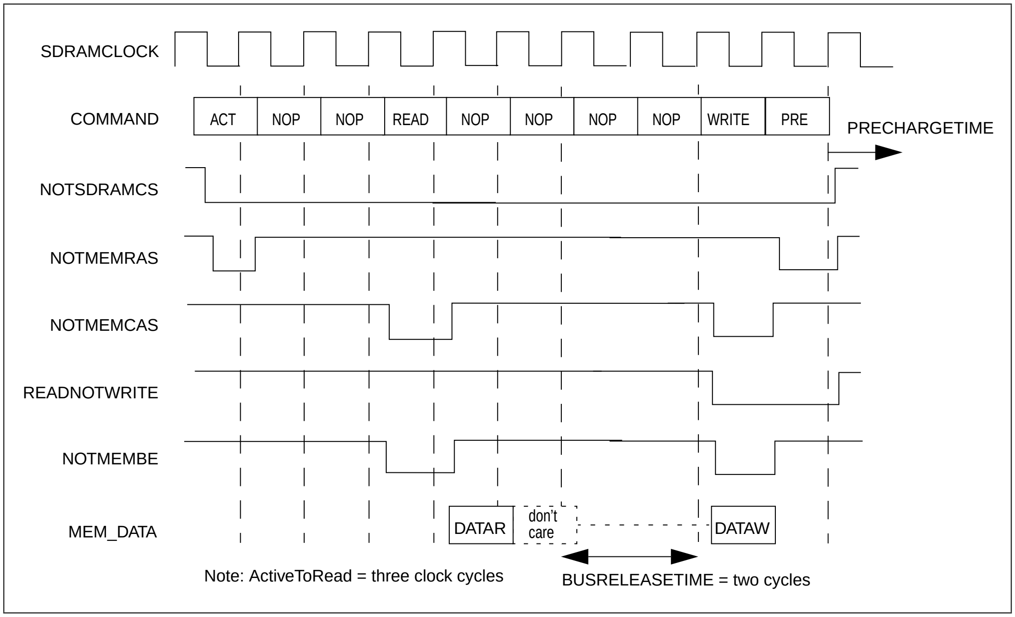


Figure 31: Page mode write after read and CAS latency = 1 cycle

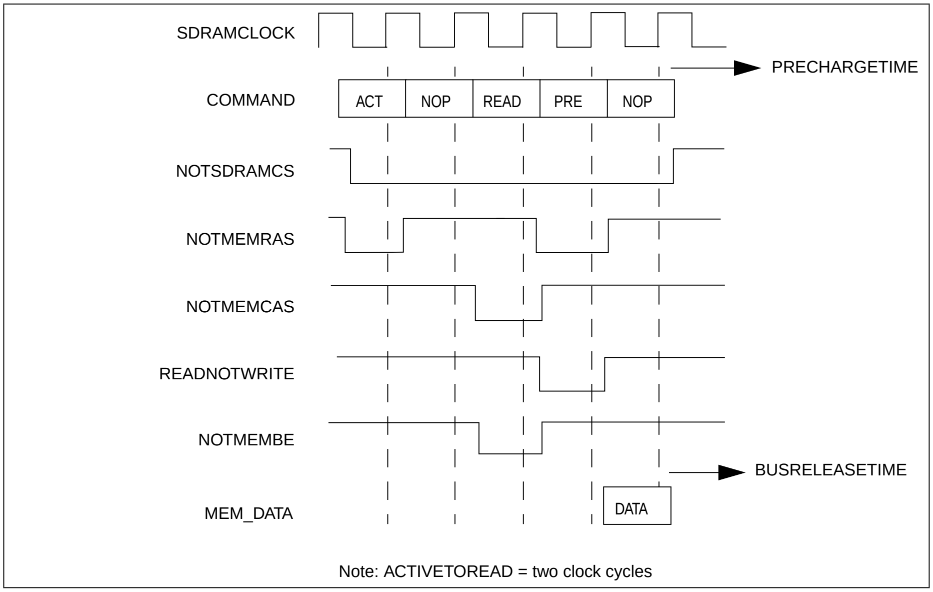


Figure 32: Single read and CAS latency = 2 cycles

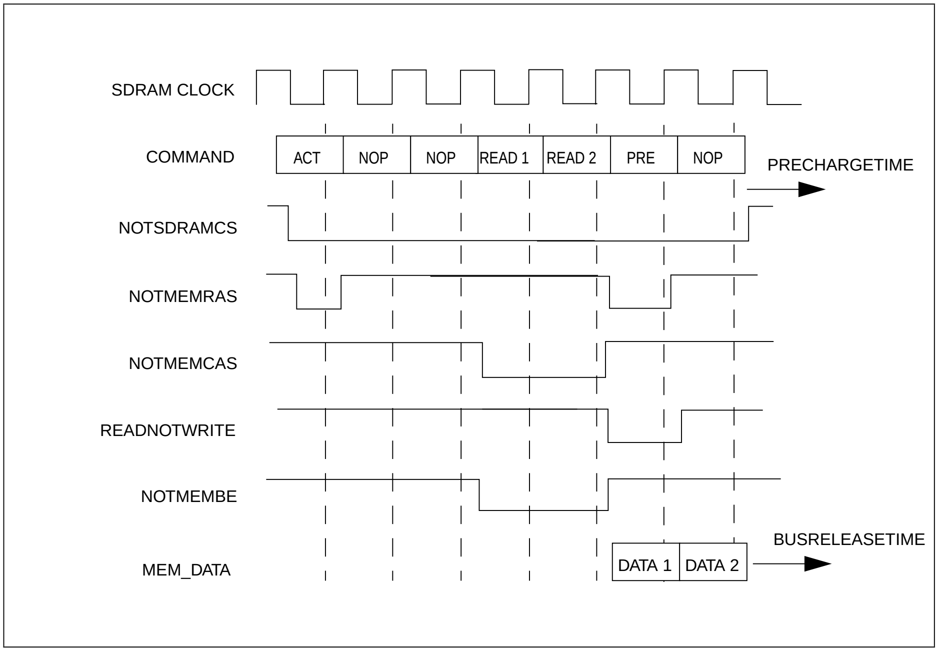


Figure 33: Page mode read and CAS latency = 2 cycles

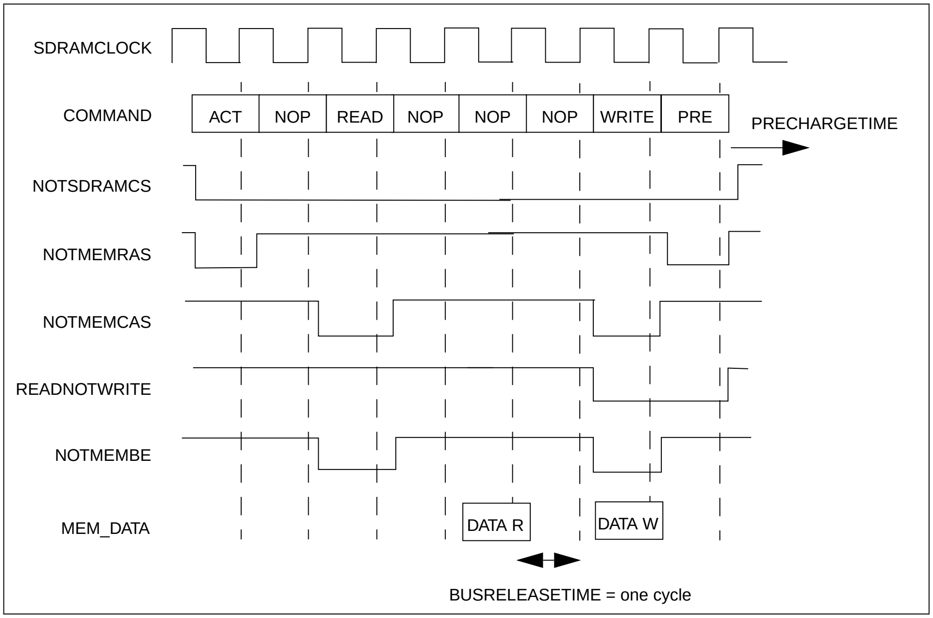


Figure 34: Page mode write after read and CAS latency = two cycles

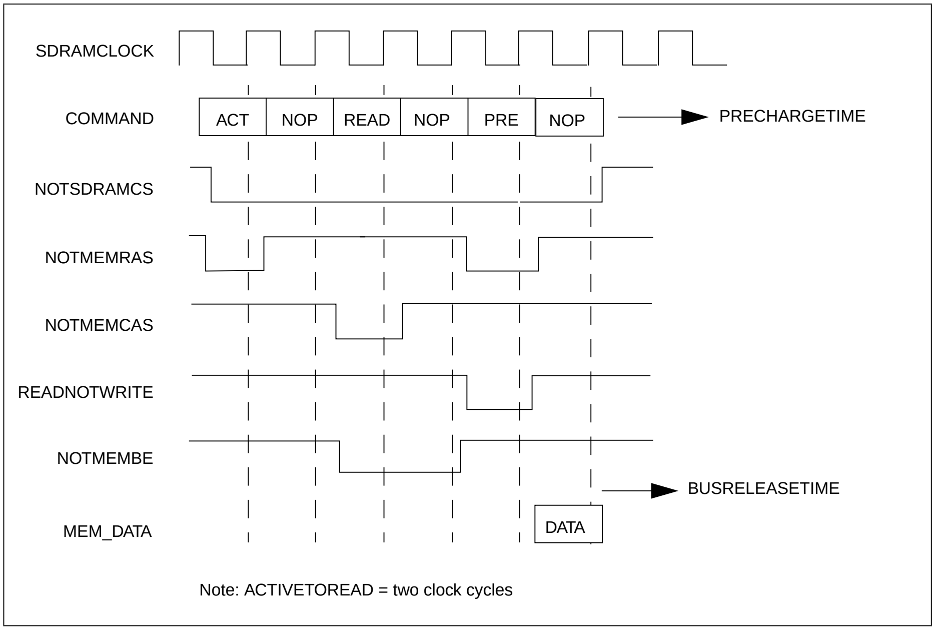


Figure 35: Single read and CAS latency = 3 cycles

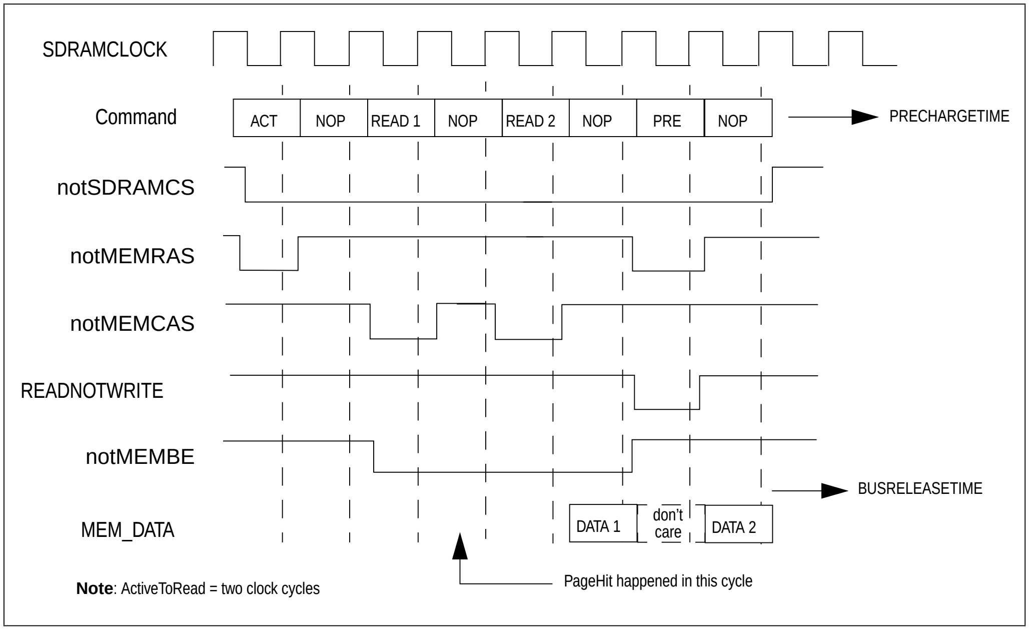


Figure 36: Page mode read and CAS latency = 3 cycles with a page hit during the NOP

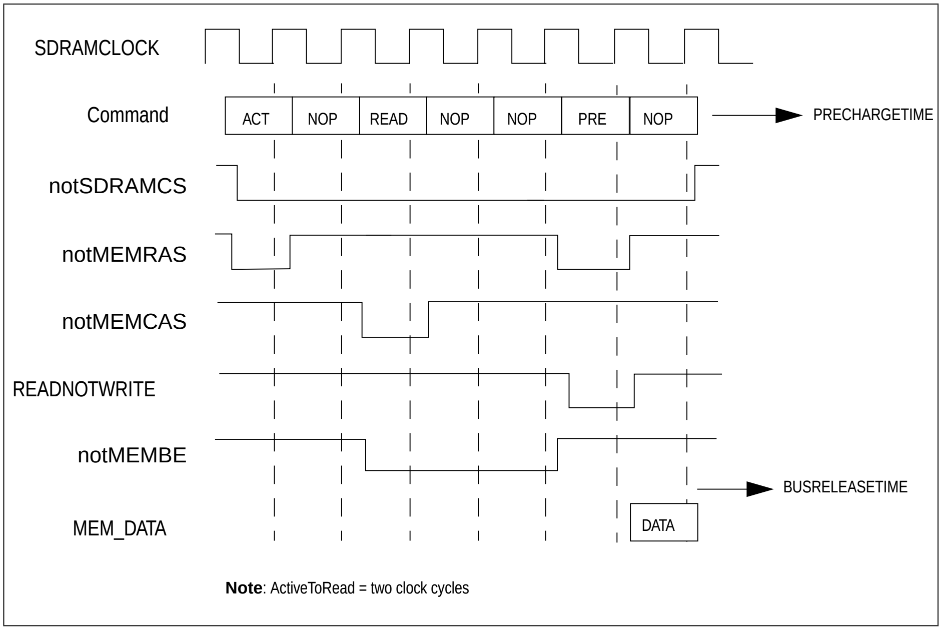


Figure 37: Single read and CAS latency = 4 cycles

2.8.9 SDRAM bank subdecoding and address selection

The SDRAM bank may be subdecoded into two or four subbanks using the EMICONFIGDATA0.SUBBANKS configuration parameter. The size and timing of each subbank must be identical. The size of each subbank is specified using the EMICONFIGDATA0.SUBBANKSIZE configuration parameter.

The address bits used to select a NOT_SDRAMCS[3:0] strobe depend on the number of subbanks, the size of the subbanks and the port size of the DRAM bank.

The SUBBANKSIZE is the total amount of memory cells contained in one SDRAM memory device. In the following table the address A means EMIADDRESS.

Number of EMI subbanks	EMI subbank size	EMI subbank selection address	Strobe selection
2	16 Mbit	A_{21}	0: NOT_SDRAMCS0 1: NOT_SDRAMCS1
	32 Mbit	A_{22}	
	64 Mbit	A_{23}	
	128 Mbit	A_{24}	
	256 MBit	A_{25}	
4	16 Mbit	A_{22} to A_{21}	00: NOT_SDRAMCS0 01: NOT_SDRAMCS1 10: NOT_SDRAMCS2 11: NOT_SDRAMCS3
	32 Mbit	A_{23} to A_{22}	
	64 Mbit	A_{24} to A_{23}	
	128 Mbit	A_{25} to A_{24}	
	256 Mbit	A_{26} to A_{25}	

Table 34: EMI subbank decoding

An address line, connected to the SDRAM, is used as a precharge mode signal. It is always address pin 10 on the SDRAM but the address line this is from on the EMI changes depending on the port width.

Port size	AP bit
8-bit	MEM_ADDRESS[10]
16-bit	MEM_ADDRESS[11]
32-bit	MEM_ADDRESS[12]

Table 35: AP bit table

Number of DRAM banks[A]	SDRAM memory size	DRAM bank selection address[B]	BS	SDRAM port size		
				32-bit	16-bit	8-bit
2	16 Mbit	A_{20}	BS0	EMI_MEM_ADDR[16]		
	32 Mbit	A_{21}				
	64 Mbit	A_{22}				
	128 Mbit	A_{23}				
	256 Mbit	A_{24}				
4	16 Mbit	A_{20} to A_{19}	BS[1:0]	EMI_MEM_ADDR[17:16]		
	32 Mbit	A_{21} to A_{20}				
	64 Mbit	A_{22} to A_{21}				
	128 Mbit	A_{23} to A_{22}				
	256 Mbit	A_{24} to A_{23}				

Table 36: SDRAM bank selection address and BS mapping with no address shifting

A. Physical SDRAM device, number of internal DRAM banks

B. Internal EMI address

Number of DRAM banks[A]	SDRAM memory size	DRAM bank selection address[B]	BS	SDRAM port size		
				32-bit	16-bit	8-bit
2	16 Mbit	A_{20}	BS0	EMI_ MEM_ ADDR [16]	EMI_ MEM_ ADDR [17]	EMI_ MEM_ ADDR [18]
	32 Mbit	A_{21}				
	64 Mbit	A_{22}				
	128 Mbit	A_{23}				
	256 Mbit	A_{24}				
4	16 Mbit	A_{20} to A_{19}	BS[1:0]	EMI_ MEM_ ADDR [17:16]	EMI_ MEM_ ADDR [18:17]	EMI_ MEM_ ADDR [19:18]
	32 Mbit	A_{21} to A_{20}				
	64 Mbit	A_{22} to A_{21}				
	128 Mbit	A_{23} to A_{22}				
	256 Mbit	A_{24} to A_{23}				

Table 37: SDRAM bank selection address and BS mapping with address shifting

A. Physical SDRAM device, number of internal DRAM banks

B. Internal EMI address

2.8.10 SDRAM refresh cycle

The SDRAM bank is periodically refreshed at intervals specified by the EMI.REFRESHINT.REFRESHINTERVAL configuration parameter. All subbanks are refreshed in the same access. After the last refresh command is issued, the EMI will wait for EMICONFIGDATA3.REFRESHTIME plus EMICONFIGDATA2.PRECHARGETIME cycles before starting a new access with a bank activate command.

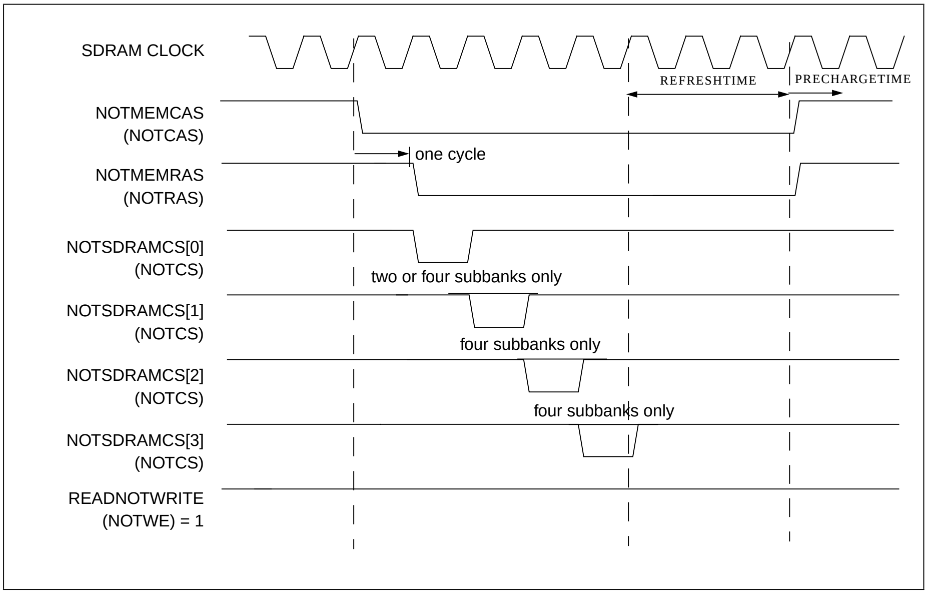


Figure 38: Generic refresh access for SDRAM bank (one SDRAM bank)

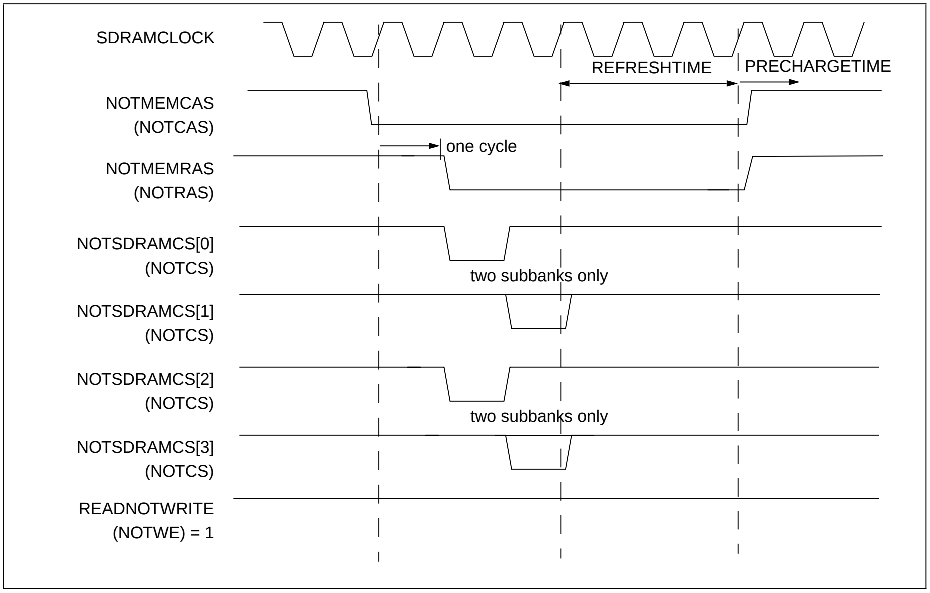


Figure 39: Generic refresh access for SDRAM bank (two SDRAM banks)

The precharge time commences after the end of the refresh access. The SDRAM bank is precharged before the refresh access starts.

Register	Parameter	Programmable value	50 MHz	100 MHz
REFRESHINT	REFRESHINTERVAL	8 to 4096 cycles	160 ns to 82 s	80ns to 41 s
BANK[N].EMICONFIGDATA2	PRECHARGETIME	1 to 16 cycles	20 ns to 320 ns	10 to 160 ns
BANK[N].EMICONFIGDATA3	REFRESHTIME	1 to 32cycles	20ns to 640 ns	10 to 320 ns

In the case of two banks configured as SDRAM, the two devices could have different configuration values for REFRESHTIME. EMI will adopt the greatest refresh time for both memories. A similar approach is adopted for precharge time.

2.8.11 Power-down

When a power-down request comes from the system, EMI goes through one of two procedures, depending on whether it is a master or slave.

EMI is a master

1 EMI completes the current access or refresh procedure if it is holding the bus. Otherwise, it gets the bus from the slave (following the usual master/slave protocol).

2 EMI does not accept any further access or bus requests.

3 The refresh counter is reset.

4 EMI jumps into the power-down state

5 The self-refresh command is sent to the SDRAMs.

6 The power-down acknowledgement is sent to the system.

7 The system can de-assert the power-down request and stop the clock.

8 When the power-down phase finishes, the system sends the clock back to EMI.

9 EMI, still in power-down mode, monitors the power-down request (deasserted), deasserts the acknowledgements and asserts the clock enable for SDRAMs.

10 EMI waits for the maximum refresh time and maximum prechargetime before serving any pending request for a new access.

EMI is a slave

In this case is up to the master to send the self-refresh command to the SDRAMs.

1 If EMI is holding the bus, it completes the access and then relinquishes the bus.

2 EMI stays in the idle state.

3 The power-down acknowledgement is sent to the system.

4 The system de-asserts the power-down request and stops the clock.

5 When the power-down phase finishes, the system sends the clock back to EMI.

6 EMI, still in the idle state, monitors the power-down request (de-asserted) and de-asserts the acknowledgements.

7 If a new request arrives, EMI will ask for the bus control, but the master will grant this request only when the SDRAMs are ready to accept a new access.

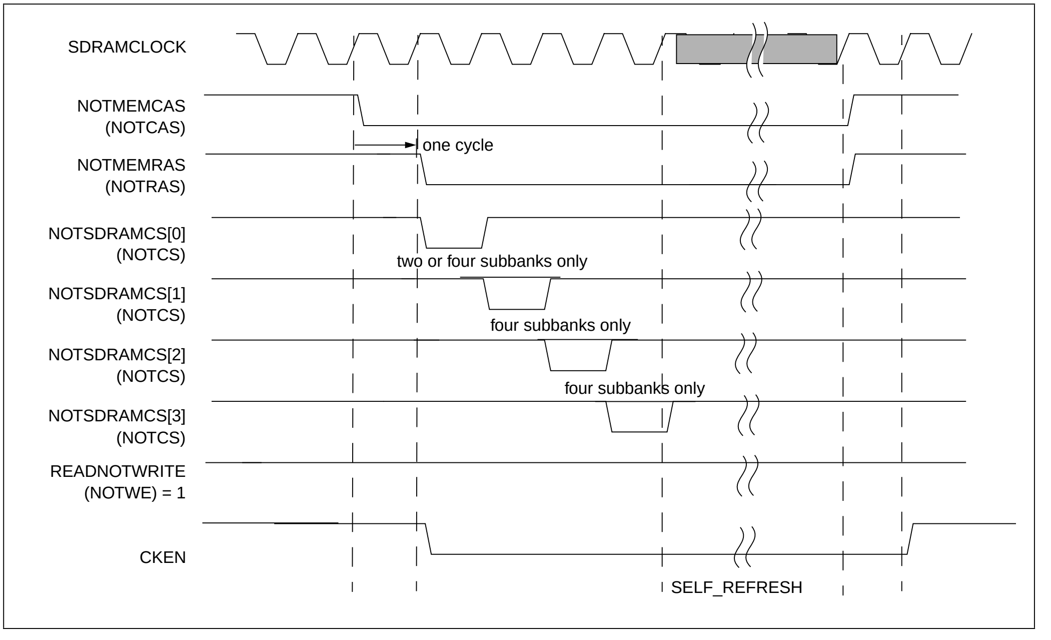


Figure 40: Self-refresh for SDRAM banks

2.8.12 Initialization

SDRAM devices require a specific initiation sequence which is achieved by accessing specific configuration registers within the EMI. This is dependent on the general SDRAM configuration and should only be begun after all other SDRAM related configuration registers have been programmed

1 The JEDEC and PC100 standards recommend the application of NOP input conditions for a minimum of 100 to 200 s after stable power and stable clock. The EMI deals with this recommendation and provides a new register called SDRAMNOPGEN. Once it is written, it will generate NOP commands to all the SDRAM devices until the SDRAMINIT register is written. The system will guarantee the maintenance of this condition for 100 to 200 s To write to this register is optional. If it is written after SDRAMINIT, the write will not have any effect.

Note: *Most SDRAMs have a power-up to first precharge, which is a minimum delay of 100 to 200 s This is a system requirement and the write to* SDRAMINIT *should not be done until this time has elapsed.*

2 Once the SDRAMINIT register is written to, and an SDRAM bank has been configured all SDRAM banks are precharged using the PRECHARGE ALL command. Eight refresh cycles are completed. Any SDRAM in the system is refreshed at the same time, as shown in *Figure 41*.

3 After the eight refresh cycles, the data written to the SDRAMMODE register is copied onto the bottom 16 address lines and a mode register set operation is executed (one or two depending upon the number of the EMI banks configured as SDRAM).

4 After the MODESETDELAY cycle delay, that is the interval between setting the mode register and executing a bank active command, the SDRAM is ready to accept an activate command. In the case of two SDRAM banks, the ModeSetDelay used comes from bank 0.

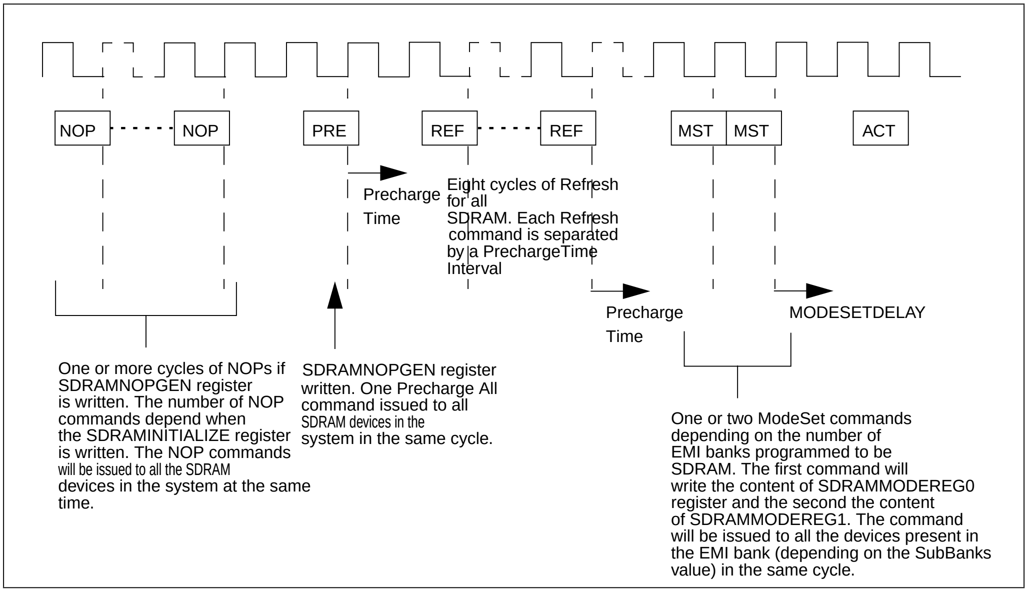

NOP
NOP
PRE
REF
REF
MST
MST
ACT
Precharge Time
Eight cycles of Refresh for all SDRAM. Each Refresh command is separated by a PrechargeTime Interval
Precharge Time
MODESETDELAY
One or more cycles of NOPs if SDRAMNOPGEN register is written. The number of NOP commands depend when the SDRAMINITIALIZE register is written. The NOP commands will be issued to all the SDRAM devices in the system at the same time.
SDRAMNOPGEN register written. One Precharge All command issued to all SDRAM devices in the system in the same cycle.
One or two ModeSet commands depending on the number of EMI banks programmed to be SDRAM. The first command will write the content of SDRAMMODEREG0 register and the second the content of SDRAMMODEREG1. The command will be issued to all the devices present in the EMI bank (depending on the SubBanks value) in the same cycle.

Figure 41: SDRAM and DRAM initialization

Following the access to EMI.SDRAMNOPGEN, devices accessed via EMI are not available until the EMI is unlocked by a subsequent access to EMI.SDRAMINIT. The access to EMI.SDRAMNOPGEN must always be followed by a subsequent access to EMI.SDRAMINIT.

The user must therefore ensure the code required for this sequence is available to the CPU in an alternate memory space such as the cache or a second memory device.

2.9 Register definition

2.9.1 EMI.STATUSCFG

EMI.STATUSCFG				0x0010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
UPDATED	[0:5]	6	Yes	Configuration updated	RO
	Operation		Shows whether the configuration registers associated with bank [n] have been updated		
	Read		Returns current value If bit [n] is set then all configuration registers associated with bank [n] have been written to at least once.		
	Write		Ignored		
	Hard reset		0x00		
-	[6:31]	26	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	HARD reset		Undefined		

Table 38: EMI.STATUSCFG

2.9.2 EMI.STATUSLOCK

EMI.STATUSLOCK				0x0018	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
LOCKED	[0:5]	6	Yes	Configuration locked	RO
	Operation		Shows whether the configuration registers are locked		
	Read		Returns current value If bit [n] is set, then all configuration registers associated with bank [n] are locked and further write accesses are ignored.		
	Write		Ignored		
	Hard reset		0x00		
-	[6:31]	26	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	Hard reset		Undefined		

Table 39: EMI.STATUSLOCK

2.9.3 EMI.LOCK

EMI.LOCK				0x0020	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PROTECT	[0:5]	6	No	Write protection	RW
	Operation		Gives write protection to the configuration registers		
	Read		Returns current value		
	Write		Updates current value If bit [n] is set then the registers EMI.BANK[N].CONFIGDATA[0:3] may only be read. Subsequent writes to these registers are ignored.		
	Hard reset		0x00		
-	[6:31]	26	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	Hard reset		Undefined		

Table 40: EMI.LOCK

2.9.4 EMI.GENCFG

EMI.GENCFG				0x0028	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
GENCFG	[0:31]	32	No	General purpose register	RW
	Operation		Propagates 32 general purpose outputs		
	Read		Returns current value		
	Write		Updates current value If bit [n] is set then the general purpose output GENCFG [n] is set.		
	Hard reset		0x00		

Table 41: EMI.GENCFG

2.9.5 EMI.SDRAMNOPGEN

EMI.SDRAMNOPGEN				0x0030	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
NOPGEN	[0]	1	No	Generate NOPs	WO
	Operation		Generates NOP commands during the initialization phase until a SDRAMInitialize is issued Used when an SDRAM is in the system		
	Read		Undefined		
	Write		Starts NOP generation		
	HARD reset		0		
-	[1:31]	31	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	HARD reset		Undefined		

Table 42: EMI.SDRAMNOPGEN

2.9.6 EMI.SDRAMMODEREG

EMI.SDRAMMODEREG				0x0038	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
REG0	[0:15]	16	No	SDRAM 0 mode register	WO
	Operation		SDRAM mode register for the SDRAM bank 0 If only one SDRAM bank is defined, it is assumed to be bank 0		
	Read		Undefined		
	Write		Updates data to be put on the bus		
	Hard reset		Undefined		
REG1	[16:31]	16	No	SDRAM 1 mode register	WO
	Operation		SDRAM mode register for the SDRAM Bank 1 If only one SDRAM bank is defined, this register is reserved		
	Read		Undefined		
	Write		Updates data to be put on the bus		
	Hard reset		Undefined		

Table 43: EMI.SDRAMMODEREG

2.9.7 EMI.SDRAMINIT

EMI.SDRAMINIT				0x0040	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
SDRAMINIT	[0]	1	No	Initialize SDRAM	WO
	Operation		Initializes any SDRAM in the system[A] This bit should be set after the setting of SDRAMMODEREG.		
	Read		Undefined		
	Write		Unitizes SDRAM		
	Hard reset		0		
-	[1:31]	31	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	Hard reset		Undefined		

Table 44: EMI.SDRAMINIT

A. If the EMI is operating in slave mode, this register should not be accessed as it is the responsibility of the bus master to initialize SDRAM and flash memory devices.

2.9.8 EMI.REFRESHINIT

EMI.REFRESHINIT				0x0048	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
REFRESHINTERVAL	[0:11]	12	No	Refresh interval setting	WO
	Operation		Defines the interval between successive refreshes in clock cycles Valid values are in the range 0x007 to 0xFFF corresponding to an interval of between 8 and 4096 cycles. Values outside this range may lead to undefined behavior.		
	Read		Undefined		
	Write		Sets the refresh interval		
	Hard reset		0x00H		
-	[12:31]	20	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	Hard reset		Undefined		

Table 45: EMI.REFRESHINIT

2.9.9 EMI.FLASHCLKSEL

EMI.FLASHCLKSEL				0x0050	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
FLASHCLOCKSELECT	[0:1]	2	No	Flash burst clock select	WO
	Operation		Sets clock ratio for burst flash clock 00: 1:1 flash operates at EMI subsystem clock 01: 1:2 flash operates at 1/2 of EMI subsystem clock 10: 1:3 flash operates at 1/3 of EMI subsystem clock 11: Reserved		
	Read		Undefined		
	Write		Sets the clock speed		
	Hard reset		10		
-	[2:31]	30	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	Hard reset		Undefined		

Table 46: EMI.FLASHCLKSEL

2.9.10 EMI.SDRAMCLKSEL

EMI.SDRAMCLKSEL				0x0058	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
SDRAMCLOCKSELECT	[0:1]	2	No	SDRAM clock select	WO
	Operation		Sets the clock ratio for SDRAM clock 00: 1:1 SDRAM operates at EMI subsystem clock 01: 1:2 SDRAM operates at 1/2 of EMI subsystem clock 10: 1:3 SDRAM operates at 1/3 of EMI subsystem clock 11: Reserved		
	Read		Undefined		
	Write		Sets the clock speed		
	Hard reset		10		
-	[2:31]	30	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	Hard reset		Undefined		

Table 47: EMI.SDRAMCLKSEL

2.9.11 EMI.MPXCLKSEL

EMI.MPXCLKSEL				0x0060	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
FLASHCLOCKSELECT	[0:1]	2	No	MPX clock select	WO
	Operation		Sets clock ratio for MPX clock		
	Read		Undefined		
	Write		Set the clock speed 00: 1:1 MPX operates at EMI subsystem clock 01: 1:2 MPX operates at 1/2 of EMI subsystem clock 10: 1:3 MPX operates at 1/3 of EMI subsystem clock 11: Reserved		
	Hard reset		10		
-	[2:31]	30	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	Hard reset		Undefined		

Table 48: EMI.MPXCLKSEL

2.9.12 EMI.CLKENABLE

EMI.CLKENABLE				0x0068	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
CLOCKENABLE	[0]	1	No	Clock enable divided clocks	WO
	Operation		Causes the flash, SDRAM and MPX clocks to be updated This operation may only occur once, further writes to this register lead to undefined behavior.		
	Read		Undefined		
	Write		1: Update flash, SDRAM and MPX clocks		
	Hard reset		00		
-	[2:31]	30	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		0		
	Hard reset		Undefined		

Table 49: EMI.CLKENABLE

2.9.13 Configuration register formats

Peripheral format

The following is a summary of the configuration register formats for peripherals.

EMI.BANK[n].EMICONFIGDATA0 (peripheral format)			RW
Bit	**Bit field**	**Function**	**Units**
[31:27]	Reserved		
26	WE_USE_OE_CONFIG	This bit must be set to 1 in case the SFlash bank (like STM58LW064A/B) requires a configurable READNOTWRITE signal for asynchronous write operation. 0: READNOTWRITE is low at the start of the access and is deactivated at the end of the access 1: WE is low following the same timing defined for OEE1TIMEWRITE and OEE2TIMEWRITE.	
25	WAITPOLARITY	Set the wait signal polarity 0: Wait active high 1: Wait active low	

Table 50: EMI.BANK[n].EMI.CONFIGDATA0 (peripheral format)

EMI.BANK[n].EMICONFIGDATA0 (peripheral format)			RW
Bit	**Bit field**	**Function**	**Units**
[24:20]	LATCHPOINT	00000: End of access cycle 00001: 1 EMI subsystem clock cycle before end of access cycle 00010: 2 EMI subsystem clock cycles before end of access cycle 00011: 3 cycles before end of access cycle 00100: 4 cycles before end of access cycle 00101: 5 cycles before end of access cycle 00110: 6 cycles before end of access cycle 00111: 7 cycles before end of access cycle 01000: 8 cycles before end of access cycle 01001: 9 cycles before end of access cycle 01010: 10 cycles before end of access cycle 01011: 11 cycles before end of access cycle 01100: 12 cycles before end of access cycle 01101: 13 cycles before end of access cycle 01110: 14 cycles before end of access cycle 01111: 15 cycles before end of access cycle 10000: 16 cycles before end of access cycle Other: Reserved	Cycles
[19:15]	DATADRIVEDELAY	0 to 31 phases	Phases
[14:11]	BUSRELEASETIME	0 to 15 cycles	Cycles
[10:9]	CSACTIVE	See *Table 27: Active code settings on page 70*	-
[8:7]	OEACTIVE	See *Table 27: Active code settings on page 70*	-
[6:5]	BEACTIVE	See *Table 27: Active code settings on page 70*	-

Table 50: EMI.BANK[n].EMI.CONFIGDATA0 (peripheral format)

EMI.BANK[n].EMICONFIGDATA0 (peripheral format)			RW
Bit	**Bit field**	**Function**	**Units**
[4:3]	PORTSIZE	00: Reserved 01: 32-bit 10: 16-bit 11: 8-bit	-
[2:0]	DEVICETYPE	001: Normal peripheral 100: Burst flash Other = Reserved	-

Table 50: EMI.BANK[n].EMI.CONFIGDATA0 (peripheral format)

EMI.BANK[n].EMICONFIGDATA1 (peripheral format)			RO
Bit	**Bit field**	**Function**	**Units**
31	CYCLENOTPHASEREAD	Change measure unit for e1/e2time accesses from phases to cycles. 0: E1TIMEWRITE and E2TIMEWRITE for CS, BE, OE expressed in system clock phases 1: E1TIMEWRITE and E2TIMEWRITE for CS, BE, OE expressed in cycles	-
[30:24]	ACCESSTIMEREAD	0: Reserved 1: Reserved 2 to 127: EMI subsystem clock cycles	Cycles
[23:20]	CSE1TIMEREAD	Falling edge of CS 0 to15: Phases or cycles after start of access cycle	Phases /Cycles
[19:16]	CSE2TIMEREAD	Rising edge of CS 0 to 15: Phases or cycles before end of access cycle	Phases /Cycles
[15:12]	OEE1TIMEREAD	Falling edge of OE 0 to 15: Phases or cycles after start of access cycle	Phases /Cycles
[11:8]	OEE2TIMEREAD	Rising edge of OE 0 to 15: Phases or cycles before end of access cycle	Phases /Cycles
[7:4]	BEE1TIMEREAD	Falling edge of BE 0 to 15: Phases or cycles after start of access cycle	Phases /Cycles
[3:0]	BEE2TIMEREAD	Rising edge of BE 0 to 15: Phases or cycles before end of access cycle	Phases /Cycles

Table 51: EMI.BANK[n].EMICONFIGDATA1 (peripheral format)

EMI.BANK[n].EMICONFIGDATA2 (peripheral format)			WO
31	CYCLENOTPHASEWRITE	Change measure unit for e1/e2time accesses from phases to cycles 0: E1TIMEWRITE and E2TIMEWRITE for CS, BE, OE expressed in system clock phases 1: E1TIMEWRITE and E2TIMEWRITE for CS, BE, OE expressed in cycles	-
[30:24]	ACCESSTIMEWRITE	0: Reserved 1: Reserved 2 to 127: EMI subsystem clock cycles	Cycles
[23:20]	CSE1TIMEWRITE	Falling edge of CS 0 to 15: Phases or cycles after start of access cycle	Phases/ Cycles[A]
[19:16]	CSE2TIMEWRITE	Rising edge of CS 0 to 15: Phases or cycles before end of access cycle	Phases/ Cycles
[15:12]	OEE1TIMEWRITE (WEE1TIMEWRITE)	Falling edge of OE 0 to 15: Phases or cycles after start of access cycle The value is used for falling edge of WE as well if EMICONFIGDATA0.WE_USE_OE_CONFIG = 1.	Phases/ Cycles
[11:8]	OEE2TIMEWRITE (WEE2TIMEWRITE)	Rising edge of OE 0 to 15: Phases or cycles before end of access cycle The value is used for rising edge of OE as well if the bit EMICONFIGDATA0.WE_USE_OE_CONFIG = 1.	Phases/ Cycles

Table 52: EMI.BANK[n].EMI.CONFIGDATA2 (peripheral format)

EMI.BANK[n].EMICONFIGDATA2 (peripheral format)			WO
[7:4]	BEE1TIMEWRITE	Rising edge of BE 0 to 15 phases or cycles after start of access cycle	Phases/ Cycles
[3:0]	BEE2TIMEWRITE	Falling edge of BE 0 to 15 phases or cycles before end of access cycle	Phases/ Cycles

Table 52: EMI.BANK[n].EMI.CONFIGDATA2 (peripheral format)

A. The value expressed in this field is interpreted in cycles rather than in phases depending on the value set for CYCLENOTPHASE

EMI.BANK[n].EMICONFIGDATA3 (peripheral format)			RW
[31:27]	Reserved		
26	STROBEONFALLING[A]	0: Strobes for burst generated on rising edge of flash clock 1: Strobes for burst generated on falling edge of flash clock	-
[25:10]	Reserved		
[9:7]	BURSTSIZE	The number of bytes which map onto the device's burst mode. 000: 2 001: 4 010: 8 011: 16 100: 32 101: 64[B] 110: 128 111: Reserved Only valid in burst mode.	-
[6:2]	DATALATENCY	The number of SFlash clock cycles between the address valid and the first data valid 00010: 2 cycles 00011: 3 cycles 00100: 4 cycles and so on until 01001: 17 cycles Others = Reserved	-
1	DATAHOLDDELAY	Extra delay when accessing same bank consecutively when in cycles between words in burst mode 0: One flash clock cycle 1: Two flash clock cycles	Cycles

Table 53: EMI.BANK[n].EMICONFIGDATA3 (peripheral format)

EMI.BANK[n].EMICONFIGDATA3 (peripheral format)			RW
0	BURSTMODE	Select synchronous flash burst mode If this bit is set only EMICONFIGDATA1.ACCESSTIMEREAD and EMICONFIGDATA3.**DATAHOLDDELAY** are relevant for strobe generation timing during read operations	-

Table 53: EMI.BANK[n].EMICONFIGDATA3 (peripheral format)

A. Configuration of EMICONFIGDATA0, EMICONFIGDATA1,EMICONFIGDATA2 relates only to the asynchronous behavior (normal peripheral and normal asynchronous behavior of flash). These registers must be programmed in terms of the EMI subsystem clock cycle. EMICONFIGDATA3 instead must be configured only if there is a burst flash and refers to the synchronous behavior of flash. The parameters in this register must be programmed in terms of flash clock cycles.

B. The 64/128 byte burst mode is due to the possible usage of the AMD device that will have a fixed 32 word burst length. The SuperHyway interface maximum transfer is 32 bytes on EMI, so in these cases the burst on flash is always interrupted.

Note: Any bit in the configuration register, which is defined as “reserved”, should be set to 0.

EMICONFIGDATA3 does not need any configuration in the case of a normal asynchronous peripheral.

The STROBEONFALLING feature of EMI means only that strobes/data/address are generated on falling edge of the SFlash clock. This DOES NOT imply that the same signal are sampled on the falling edge by the memories. The EMI assume memory will always sample on rising edge anyway. The STROBEONFALLING feature has been implemented only to possibly extend the HOLD time of half a cycle to help padlogic implementation.

MPX format

The following is a summary of the configuration register formats for MPX. Any bit in the configuration registers which is defined as reserved should be set to 0.

EMI.BANK[n].EMICONFIGDATA0 (MPX format)			RW
Bits	**Bit field**	**Function**	**Units**
[31:27]	Reserved		
26	STROBEONFALLING	0: Strobes, data, address for MPX generated on rising edge of the MPX clock 1: Strobes, data, address for MPX generated on falling edge of the MPX clock	-
25	WAITPOLARITY	Set the Wait input pin polarity: 0: Wait active high 1: Wait active low	-
[24:14]	Reserved		
13	EXTENDEDMPX	When this bit is set, the MPX interface uses ST MPX super-set opcodes (1X0: 16 bytes transfer), otherwise the standard set is used.	-
[12:11]	BUSRELEASETIME	Specifies time needed to release the bus for MPX agent 00: One MPX clock cycle 01: Two cycles 10: Three cycles 11: Four cycles	-
[10:9]	WAITSTATESFRAME	Specifies internal wait to be inserted for accesses (read or write) after the first 00: No wait states 01: One wait state 10: Two wait states 11: Three wait states	Cycles

Table 54: EMI.BANK[n].EMICONFIGDATA0 (MPX format)

EMI.BANK[n].EMICONFIGDATA0 (MPX format)			RW
Bits	**Bit field**	**Function**	**Units**
[8:6]	WAITSTATESREAD	Specifies internal wait to be inserted for first read 000: 0 wait state 001: One wait state 010: Two wait states 0 11: Three wait states 100: Four wait states 101: Five wait states 110: Six wait states 111: Seven wait states	Cycles
[5:3]	WAITSTATESWRITE	Specifies internal wait to be inserted for first write 000: 0 wait state 001: One wait state 010: Two wait states 0 11: Three wait states 100: Four wait states 101: Five wait states 110: Six wait states 111: Seven wait states	Cycles
[2:0]	DEVICETYPE	Sets the format of the config register 011: MPX	-

Table 54: EMI.BANK[n].EMICONFIGDATA0 (MPX format)

EMI.BANK[n].EMICONFIGDATA1 (MPX format)			Res
Bits	**Bit field**	**Function**	**Units**

Table 55: EMI.BANK[n].EMICONFIGDATA1 (MPX format)

EMI.BANK[n].EMICONFIGDATA1 (MPX format)		Res
[0:31]	Reserved	

Table 55: EMI.BANK[n].EMICONFIGDATA1 (MPX format)

EMI.BANK[n].EMICONFIGDATA2 (MPX format)			Res
Bits	**Bit field**	**Function**	**Units**
[0:31]	Reserved		

Table 56: EMI.BANK[n].EMICONFIGDATA2 (MPX format)

EMI.BANK[n].EMICONFIGDATA3 (MPX format)			Res
Bits	**Bit field**	**Function**	**Units**
[0:31]	Reserved		

Table 57: EMI.BANK[n].EMICONFIGDATA3 (MPX format)

SDRAM format

The following are the configuration registers formats for SDRAM. Any bit in the configuration registers which is defined as reserved should be set to 0.

EMI.BANK[n].EMICONFIGDATA0 (SDRAM format)			RW
Bit	**Bit field**	**Function**	**Units**
[31:27]	Reserved		
26	STROBEONFALLING	0: Strobes, data and address for SDRAM generated on rising edge of the clock (recommended mode for SDRAM) 1: Strobes, data and address for SDRAM generated on falling edge of the clock If two EMI banks are configured as SDRAM, the STROBESONFALLING bit must be set the same for both banks.	-
[25:15]	Reserved		
[14:13]	BUSRELEASETIME	One to four SDRAM clock cycles	Cycles
[12:11]	SUBBANKS	00: One 01: Two 10: Four 11: Reserved	-

Table 58: EMI.BANK[n].EMICONFIGDATA0 (SDRAM format)

EMI.BANK[n].EMICONFIGDATA0 (SDRAM format)			RW
Bit	**Bit field**	**Function**	**Units**
[10:8]	SUBBANKSIZE	The physical size of the SDRAM device used in each EMI bank or subbank. Even if one subbank is selected in EMI_SUBBANKS (bits12:11), the subbank size must be set to match the physical size of the SDRAM device used on this bank. 000: 16 Mbit 001: 32 Mbit 010: 64 Mbit 011: 128 Mbit 100: 256Mbit 101, 110, 111: Reserved	-
[7:5]	SHIFTAMOUNT	Column address width 000: 7 001: 8 010: 9 011: 10 100: 11 101: 12 110: 13 111: 14	-
[4:3]	PORTSIZE	00: Reserved 01: 32 bit 10: 16 bit 11: 8 bit	-
[2:0]	DEVICETYPE	010: SDRAM. Sets the format of the config register	-

Table 58: EMI.BANK[n].EMICONFIGDATA0 (SDRAM format)

EMI.BANK[n].EMICONFIGDATA1 (SDRAM format)			RW
[31:23]	Reserved		
[22:0]	RASBITS	Page address mask for address bits [29:7]. For a 32-bit data bus port width, memory address bits [7:1] of the internal memory address are assumed to be 0 by default. For example for a 32-bit data bus (8-bit SDRAM column size) set RASBITS[2:0] to 0 and RASBITS[22:3] to 1. For a 16-bit data bus port width, memory address bits [6:1] of the internal memory address are assumed to be 0 by default. For example for a 16-bit data bus (8-bit SDRAM column size) set RASBITS[1:0] to 0 and RASBITS[22:2] to 1.	-

Table 59: EMI.BANK[n].EMICONFIGDATA1 (SDRAM format)

EMI.BANK[n].EMICONFIGDATA2 (SDRAM format)			RW
[31:8]	Reserved		
[7:4]	PRECHARGETIME	1 to 16 SDRAM clock cycles	Cycles
[3:0]	Reserved		

Table 60: EMI.BANK[n].EMI.EMICONFIGDATA2 (SDRAM format)

EMI.BANK[n].EMICONFIGDATA3 (SDRAM format)			RW
[31:20]	Reserved		
[19:18]	MODESETDELAY	3 to 6 SDRAM clock cycles	Cycles
[17:13]	REFRESHTIME	1 to 32 SDRAM clock cycles	Cycles
[12:10]	ACTIVATETOREAD	1 to 8 SDRAM clock cycles	Cycles
[9:7]	ACTIVATETOWRITE	1 or 8 SDRAM clock cycles	Cycles

Table 61: EMI.BANK[n].EMICONFIGDATA3 (SDRAM format)

EMI.BANK[n].EMICONFIGDATA3 (SDRAM format)			RW
[6:4]	CASLATENCY	One to eight SDRAM clock cycles	Cycles
3	DRAM BANKS	0: Two DRAM banks 1: Four DRAM banks	-
[2:0]	WRITERECOVERYTIME	000: Reserved 001: One SDRAM clock cycle 010: Two SDRAM clock cycles 011: Three SDRAM clock cycles 100: Four SDRAM clock cycles 101: Five SDRAM clock cycles 110: Six SDRAM clock cycles 111: Seven SDRAM clock cycles	Cycles

Table 61: EMI.BANK[n].EMICONFIGDATA3 (SDRAM format)

3 EMI buffer

3.1 Overview

The EMI buffer is located between the SuperHyway and the EMI. It handles the regular exchange of information between these two blocks, buffering all data from the SuperHyway interconnect before passing it on to the EMI block. In particular the block performs the functions listed below.

- Guarantees a continuous data flow to the EMI block.

  This means that the block must be able to transmit a packet from the STBus interconnect block towards the EMI block without a gap between two consecutive cells in the packet (supports EMI-MPX interface functionality).

- Support for EMI bank size programmability as well as the generation of the right external bank to access.

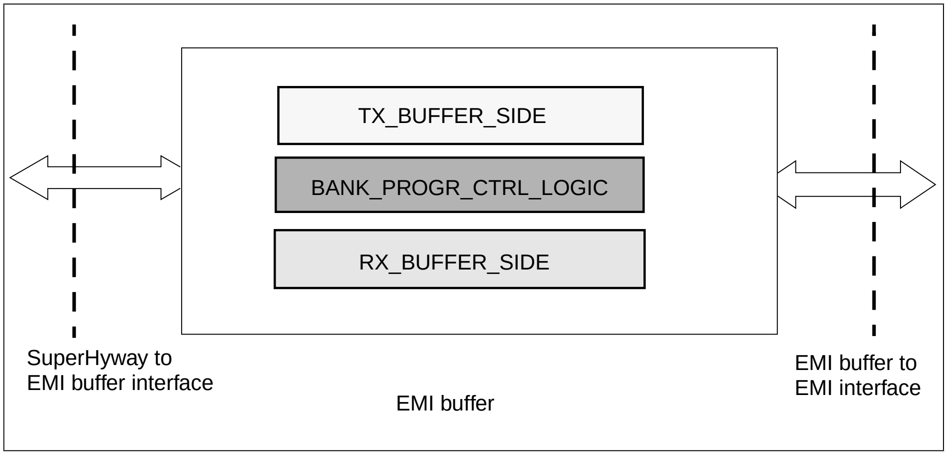

TX_BUFFER_SIDE
BANK_PROGR_CTRL_LOGIC
RX_BUFFER_SIDE
SuperHyway to EMI buffer interface
EMI buffer
EMI buffer to EMI interface

Figure 42: EMI buffer

The TX_BUFFER_SIDE handles the request packet from the SuperHyway to the EMI block. The EMI buffer deals with the timing constraints and continuous data flow in a burst transaction.

The RX_BUFFER_SIDE completes the same operation as the TX_BUFFER_SIDE for the response packet request from the EMI block to the SuperHyway block.

The BANK_PROGR_CTRL_LOGIC controls the internal registers of the EMI buffer block containing the top address of the external memory banks. This block also provides the total number of external memory banks enabled at the same time (after setting an internal register).

3.2 Register address map

The EMI buffer block is characterized by the following seven internal registers:

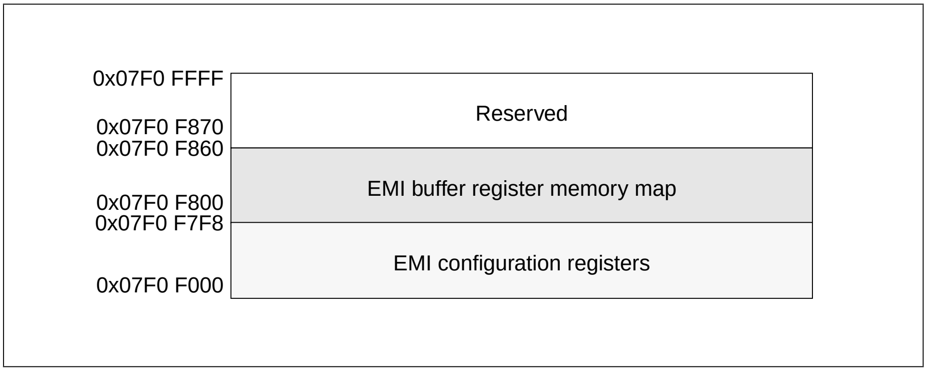
- six related to the accessible external memory banks (each composed of 8 bits),
- one related to the value of the total number of banks registers enabled at the same time (composed of 3 bits).

Table 62: Register summary table

Register name	Description	Address offset	Type
BANK_0_BASE_ADDRESS	External memory bank 0 base address bits 27 to 22, see *BANK_0_BASE_ADDRESS on page 154*	0x000	RW
BANK_1_BASE_ADDRESS	External memory bank 1 base address bits 27 to 22, see *BANK_0_BASE_ADDRESS on page 154*	0x010	RW
BANK_2_BASE_ADDRESS	External memory bank 2 base address bits 27 to 22, see *BANK_0_BASE_ADDRESS on page 154*	0x020	RW
BANK_3_BASE_ADDRESS	External memory bank 3 base address bits 27 to 22, see *BANK_0_BASE_ADDRESS on page 154*	0x030	RW
BANK_4_BASE_ADDRESS	External memory bank 4 base address bits 27 to 22, see *BANK_0_BASE_ADDRESS on page 154*	0x040	RW
BANK_5_BASE_ADDRESS	External memory bank 5 base address bits 27 to 22, see *BANK_0_BASE_ADDRESS on page 154*	0x050	RW
BANK_ENABLED	Total number of enabled banks, see *BANK_0_BASE_ADDRESS on page 154*	0x060	RW

3.2.1 EMI memory map

Figure 43: EMI memory map



3.2.2 Register descriptions

All the registers described in this section are nonvolatile.

BANK_0_BASE_ADDRESSExternal memory bank 0 top address

31–6	5–0
Reserved	BANK_0_BASE_ADDRESS

Address:	*EMIBufferBaseAddress* + 0x000
Type:	Read/Write
Reset:	0x00

Description

Contains bits 27 to 22 of the base address of external memory bank 0. Accesses to this address space cause transfer on EMI bank 0.

Note: Do not change from the reset state when booting from ROM.

BANK_1_BASE_ADDRESS External memory bank 1 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										BANK_1_BASE_ ADDRESS					

Address:	*EMIBufferBaseAddress* + 0x010
Type:	Read/Write
Reset:	0x04

Description

Contains bits 27 to 22 of the base address of external memory bank 1. Accesses to this address space cause transfer on EMI bank 1.

BANK_2_BASE_ADDRESS External memory bank 2 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										BANK_2_BASE_ ADDRESS					

Address:	*EMIBufferBaseAddress* + 0x020
Type:	Read/Write
Reset:	0x08

Description

Contains bits 27 to 22 of the base address of external memory bank 2. Accesses to this address space cause transfer on EMI bank 2.

BANK_3_BASE_ADDRESS External memory bank 3 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										BANK_3_BASE_ ADDRESS					

Address:	*EMIBufferBaseAddress* + 0x030
Type:	Read/Write
Reset:	0x0C

Description

Contains bits 27 to 22 of the base address of external memory bank 3. Accesses to this address space cause transfer on EMI bank 3.

BANK_4_BASE_ADDRESS External memory bank 4 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										BANK_4_BASE_ ADDRESS					

Address:	*EMIBufferBaseAddress* + 0x040
Type:	Read/Write
Reset:	0x10

Description

Contains bits 27 to 22 of the base address of external memory bank 4. Accesses to this address space cause transfer on EMI bank 4.

BANK_5_BASE_ADDRESS External memory bank 5 base address

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										BANK_5_BASE_ADDRESS					

Address:	*EMIBufferBaseAddress* + 0x050
Type:	Read/Write
Reset:	0x14

Description

Contains bits 27 to 22 of the base address of external memory bank 5. Accesses to this address space cause transfer on EMI bank 5.

BANK_ENABLED Enabled bank 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
Reserved																													BANKS_ENABLED		

Address:	*EMIBufferBaseAddress* + 0x060
Type:	Read/Write
Reset:	0x06

Description

Contains the total number of bank registers enabled. At reset all the banks are enabled.

6 (110) = All banks enabled	5 (101) = Banks 4 to 0 enabled
4 (100) = Banks 3 to 0 enabled	3 (011) = Banks 2 to 0 enabled
2 (010) = Banks 0 and 1 enabled	1 (001) = Bank 0 only enabled

When the number of banks is reduced by the BANKS_ENABLED register, the last bank (that is the bottom bank) takes its own area plus the remaining area of the banks disabled. For example if only five banks are enabled, bank 5 is disabled, then the bank 4 region contains its own area plus the bank 5 area.

4 PCI bus interface (PCI)

4.1 Introduction

The ST40 integrates a fully featured PCI 2.1 compliant bus interface. This chapter describes that interface and how it appears to the rest of the system.

This PCI bus controller provides a bridge between the SuperHyway and the PCI bus. The main goals of the controller are:

- to provide a channel for the CPU to access and control PCI systems on the external PCI bus,
- to provide a channel for PCI devices to access the address space of SH-4/ST40, particularly the system memory through the EMIs,
- to provide the ability for the ST40 to behave as an intelligent add-in card in a PCI system in which another processor is the host.

In most situations the ST40 PCI bridge will act as a host bridge with the ST40 CPU as the host, this is described as host mode. The remainder of this chapter will assume host mode, except where noted. There are situations where the ST40 and future ST40 based designs in which the PCI bridge will act as a peripheral PCI device in a system where another processor is the host (for example a PC.). This is known as satellite mode.

The ST40 system architecture supports a hierarchical interconnect structure. The structure of particular implementations may allow for concurrent data streams to use the interconnect.

Some knowledge of the PCI 2.1 specification is assumed in this chapter; see *Section 4.7: References on page 280*. The focus for this description is the ST40 implementation.

4.1.1 Features

The PCI bus interface has the following features:

- 0 to 66 MHz operation,
- 32-bit data path,
- PCI master and target functions,
- host bridge mode and satellite mode,
- PCI arbiter (in host mode) supporting four external masters,
- configuration generation by PCI configuration mechanism number 1 (in host mode),
- burst transfers,
- parity check and error reporting,
- exclusive access, target: once locked only PCI access to ST40 is from lock owner[1],
- PCI clock is asynchronous to ST40 clock,
- host bridge mode and satellite mode.

The PCI bus controller supports applications which seek compliance with the power management interface specification. In particular, the necessary configuration registers are provided along with a supporting interrupt to the ST40 processor to indicate that the registers have been written by the host processor.

1. The lock only prevents other PCI initiators from accessing the ST40RA. The lock is not supported internal to the ST40RA so that software has the responsibility to ensure that other subsystems do not break the semantics of the lock.

PCI features not supported

The PCI bus interface does not support:

- external cache support (no NOT_SBO or SDONE pins),
- PCI JTAG,
- dual address cycles,
- interrupt acknowledge,
- special cycles.

4.1.2 Supported PCI commands

C/BE[3:0]	Command type	PCI master	PCI target
0000	Interrupt acknowledge	8	Ignore
0001	Special cycle	8	Ignore
0010	I/O read	4	4
0011	I/O write	4	4
0100	Reserved	8	Ignore
0101	Reserved	8	Ignore
0110	Memory read	4	4
0111	Memory write	4	4
1000	Reserved	8	Ignore
1001	Reserved	8	Ignore
1010	Configuration read	✔(host mode)	4
1011	Configuration write	✔(host mode)	4
1100	Memory read multiple	4	4
1101	Dual address cycle	8	Ignore
1110	Memory read line	4	✔(memory read)
1111	Memory write and invalidate	8	✔(memory write)

4.1.3 Signal description

Figure 44 illustrates the PCI signals used by the ST40 interface. A complete functional description of the pin list is given in *Section 4.6: Pin list on page 278*.

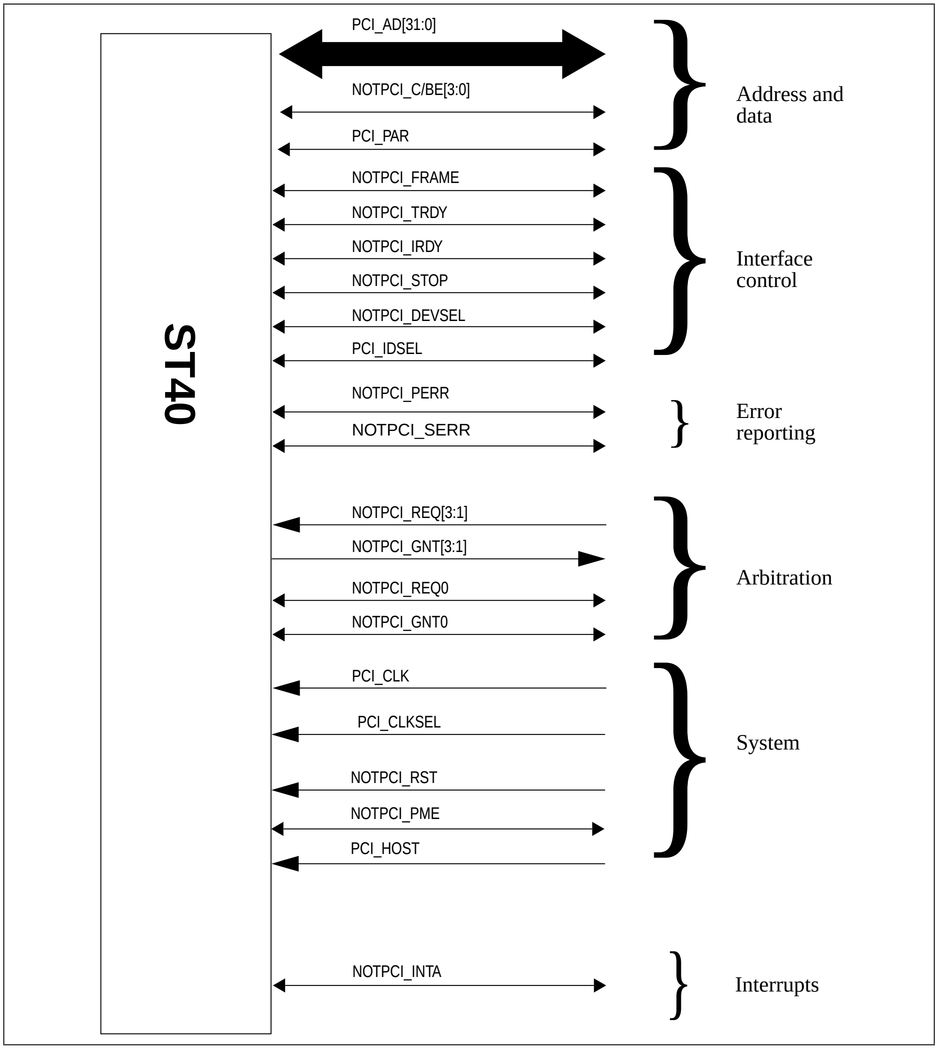


Figure 44: PCI bus pin summary

4.1.4 Example configurations

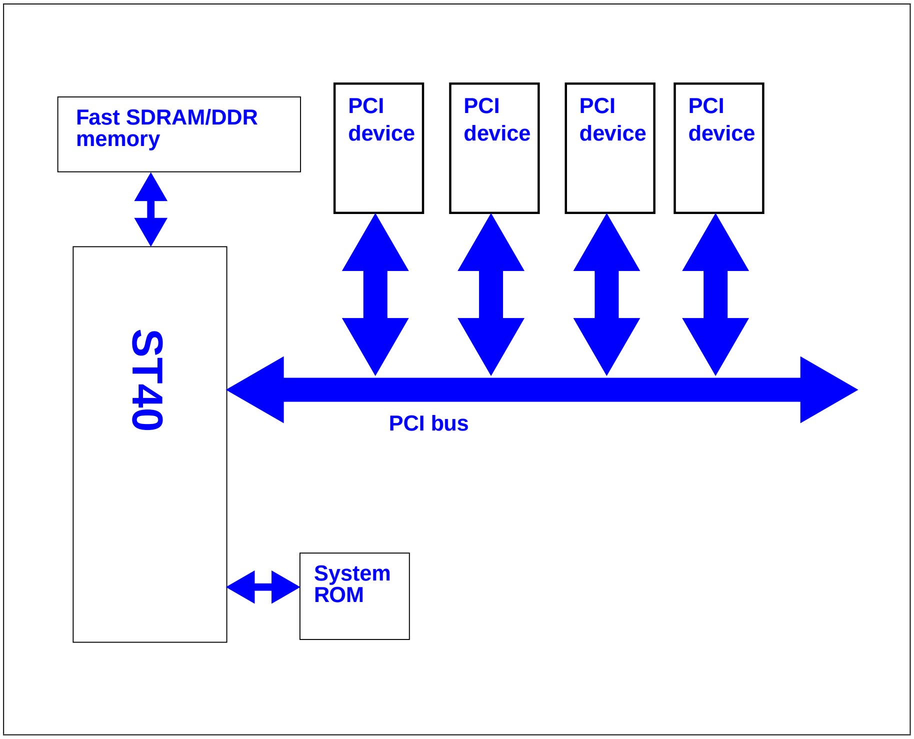


Figure 45: Example host mode system

Figure 45 shows an application diagram of the ST40 system used in host mode.

- The ST40 is the host CPU of the PCI system.
- Software running on the ST40 is responsible for configuring all PCI devices.
- The ST40 ST40RA PCI controller may be configured to allow PCI devices to have access to a window of the ST40's local memory space.
- The ST40 is responsible for dealing with interrupts and errors.

.

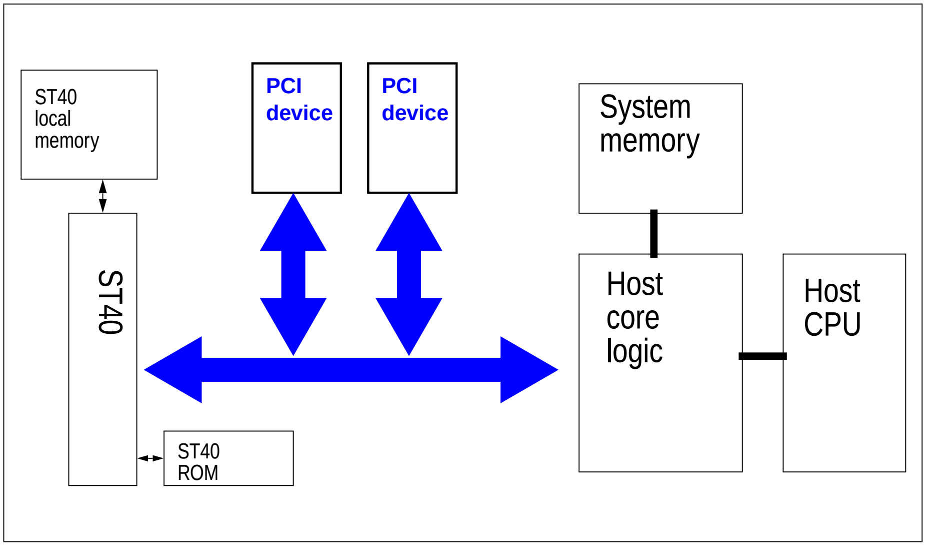


Figure 46: Example satellite mode system

Figure 46 illustrates an application diagram of the ST40 used in satellite mode.

- The ST40 acts as an embedded controller of an add-in card.
- Software running on the host CPU is responsible for configuring all PCI devices including the ST40 PCI controller.
- The host core logic may allow the ST40 to access system memory.
- The host and or other PCI devices may be allowed access to the ST40's local memory or the ST40 can use this as private memory.
- The host core logic is responsible for PCI bus arbitration and for dealing with interrupts and errors.

4.1.5 Basic operation

Essentially the PCI controller module is a bridge between two different bus systems. Its main responsibility is mapping address spaces between the local SuperHyway domain and the PCI bus domain to allow bi-directional access. In addition the PCI controller module has a number of other responsibilities including mapping interrupts, errors and, depending on the mode, it may be required to arbitrate and support configuration of devices on the bus.

4.2 Local address map

Table 63 shows the PCI module address map. All addresses are given as offsets from the module's base address. See the system address map for details.

Area	Block type	Offset	Address space size
PCI memory	DB	0x0000 0000 0x05FF FFFF	96 Mbytes
PCI memory	DB	0x0000 0000 0x05FF FFFF	96 Mbytes
PCI I/O	DB	0x0600 0000 0x06FF FFFF	16 Mbytes
Registers space	CB	0x0700 0000 0x07FF FFFF	16 Mbytes

Table 63: PCI interface memory map

Accesses to the local PCI memory area will be translated into PCI memory commands on the PCI bus. Accesses to the local PCI I/O area will be translated into PCI I/O commands on the PCI bus.

The register space for the PCI module is partitioned into two banks as shown in *Table 64*. A list of all the registers is provided in *Section 4.5: Registers on page 184*

Register bank	Format
PCI local	Implementation specific
PCI configuration space registers (CSR)	PCI 2.1 configuration header type 0

Table 64: PCI register banks

4.3 Transactions

The PCI module is involved in two classes of PCI transactions.

- Transactions in which the local system is the initiator and a device on the PCI bus is the target. These are called PCI master transactions and are described in *Section 4.3.1*.
- Transactions in which a device on the PCI bus is the initiator and the local PCI interface is the target. These are called PCI target transactions and are described in *Section 4.3.2*.

4.3.1 PCI master transactions

When the PCI bus module is the initiator of PCI bus transactions the controller allows:

- access to PCI memory space by mapping part of the ST40 local address space into the PCI memory space,
- access to PCI I/O space by mapping part of the ST40 local address space into the PCI I/O space,
- access to the PCI configuration space by using a pair of PCI local bank registers to specify the address and data for a PCI configuration space access, using a procedure which conforms to PCI local bus specification 2.1 mechanism 1.

These mappings are illustrated by *Figure 47*.

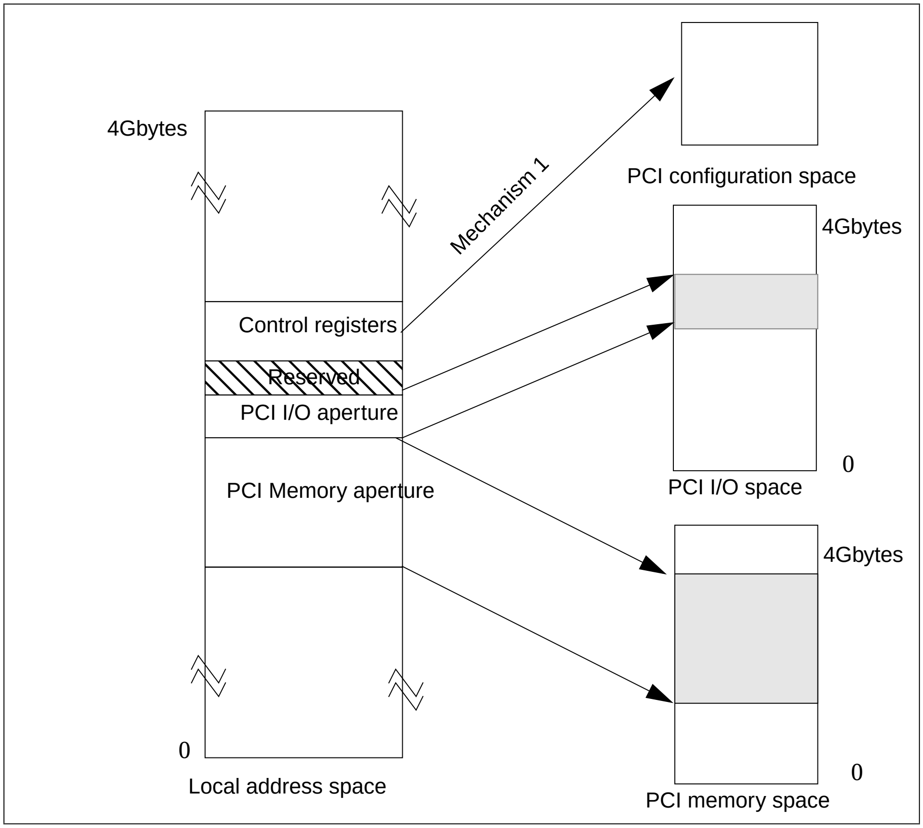


Figure 47: PCI master address space

Unsupported PCI commands as master

PCI master transactions do not support the following commands:

- IACK,
- dual address cycle,
- memory write and invalidate,
- special cycle.

Memory read/write

Local accesses which fall into the PCI memory area are translated into PCI memory transactions. The lower 16 bits of the local address are mapped unchanged to the PCI address. The upper 16 bits of the local address are translated using a combination of the PCI memory space bank register (PCI.MBR) and the PCI memory space bank mask register (PCI.MBMR).

PCI.MBMR allows the selection of bits from either the local address, or PCI.MBR. Where a bit in PCI.MBMR is 1, the corresponding bit of the local address is copied to the generated PCI address. Where the PCI.MBMR bit is 0, the corresponding bit of PCI.MBR is copied.

Only certain bit patterns are allowed in PCI.MBMR. A side-effect of the algorithm used is that the value placed in PCI.MBMR also controls the PCI memory aperture size. See *Section : PCI memory space bank register on page 229* and *Section : PCI memory space bank mask register on page 233*.

Example 1

Example 1 gives a unique address mapping for the entire 96-Mbyte address range from 0xB000 0000 to 0xB5FF FFFF.

PCI.MBR = 0x1000 0000

PCI.MBMR = 0x07FF 0000

Local address (96 Mbyte aperture)	PCI address (96 Mbyte aperture)
0xB000 0000	0x1000 0000
0xB321 DC6C	0x1321 DC6C
0xB5FF FFFF	0x15FF FFFF

Table 65: Example mappings giving a 96-Mbyte PCI memory address range

Example 2

In example two the aperture size is restricted to 64 Kbytes. Local addresses at 64-Kbyte intervals generate the same PCI address.

PCI.MBR = 0x1111 0000

PCI.MBMR = 0x0000 0000.

Local address (96 Mbyte aperture)	PCI address (64 kbyte aperture)
0xB000 0000	0x1111 0000
0xB000 FFFF	0x1111 FFFF
0xB001 0000	0x1111 0000
0xB321 DC6C	0x1111 DC6C
0xB5FF FFFF	0x1111 FFFF

Table 66: Address mapping for 64-Kbyte aperture

I/O read/write

Local accesses which fall into the PCI I/O area are translated into PCI I/O transactions. The lower 16 bits of the local address are mapped unchanged to the PCI address. The upper 16 bits of the local address are translated using a combination of the PCI I/O space bank register (PCI.IOBR) and the PCI I/O space bank mask register (PCI.IOBMR).

PCI.IOBMR allows the selection of bits from either the local address, or PCI.IOBR. Where a bit in PCI.IOBMR is 1, the corresponding bit of the local address is copied to the generated PCI address. Where the PCI.IOBMR bit is 0, the corresponding bit of PCI.IOBR is copied.

Only certain bit patterns are allowed in PCI.IOBMR. A side-effect of the algorithm used is that the value placed in PCI.IOBMR also controls the PCI I/O aperture size. See *Section : PCI I/O space bank register on page 230* and *Section : PCI I/O space bank mask register on page 235*.

Example 1

Example 1 gives a unique address mapping for the entire 16 Mbytes address range from 0xB6000000 to 0xB6FFF FFFF.

PCI.MBR = 0x1000 0000

PCI.MBMR = 0x00FF 0000

Local address (16 Mbyte aperture)	PCI address (16 Mbyte aperture)
0xB600 0000	0x1600 0000
0xB6FF FFFF	0x16FF FFFF

Table 67: Example mappings giving a 16-Mbyte PCI I/O address range

Example 2

In example two the aperture size is restricted to 64 Kbytes. Local addresses at 64-Kbyte intervals generate the same PCI address.

PCI.IOBR = 0x1111 0000

PCI.IOBMR = 0x0000 0000

Local address (16 Mbyte aperture)	PCI address (16 Kbyte aperture)
0xB600 0000	0x1111 0000
0xB600 FFFF	0x1111 FFFF
0xB601 FFFF	0x1111 0000
0xB621 DC6C	0x1111 DC6C
0xB6FF FFFF	0x1111 FFFF

Table 68: Example mappings giving a 64-Kbyte PCI I/O address range

Configuration read/write

The PCI controller supports configuration mechanism 1 similar[1] to that defined in the PCI local bus specification 2.1 (for PC-AT compatible systems) by providing configuration address and data port registers. The register PCI.PAR which is specified as the CONFIG_ADDRESS register and PCI.PDR is specified as the CONFIG_DATA register in the standard. The general mechanism for accessing configuration space is to write a value into the PCI.PAR register which specifies the PCI bus, device on that bus and configuration register in that device being accessed. A subsequent read or write to PCI.PDR will then cause the PCI controller to translate the PCI.PAR value to the requested configuration cycle on the PCI bus.

4.3.2 PCI target transactions

When the ST40's PCI controller is the target of PCI local bus transactions the controller allows:

- access to ST40RA local memory space (including both PCI bank registers) from PCI devices using the PCI memory read/write commands,
- access to the PCI configuration registers using PCI configuration commands,
- access to the PCI configuration registers using PCI I/O commands.

Unsupported PCI commands as target

PCI target transactions do not support the following commands:

- I/O read/write to local address space,
- type 1 configuration read/write,
- special cycle,
- IACK cycle,
- dual address cycles.

1. The dIfferences are that the ST40 mechanism maps the register pair into local memory space and the bottom two bits of the address register are made read/ write giving direct software access to type 0 and type 1 accesses.

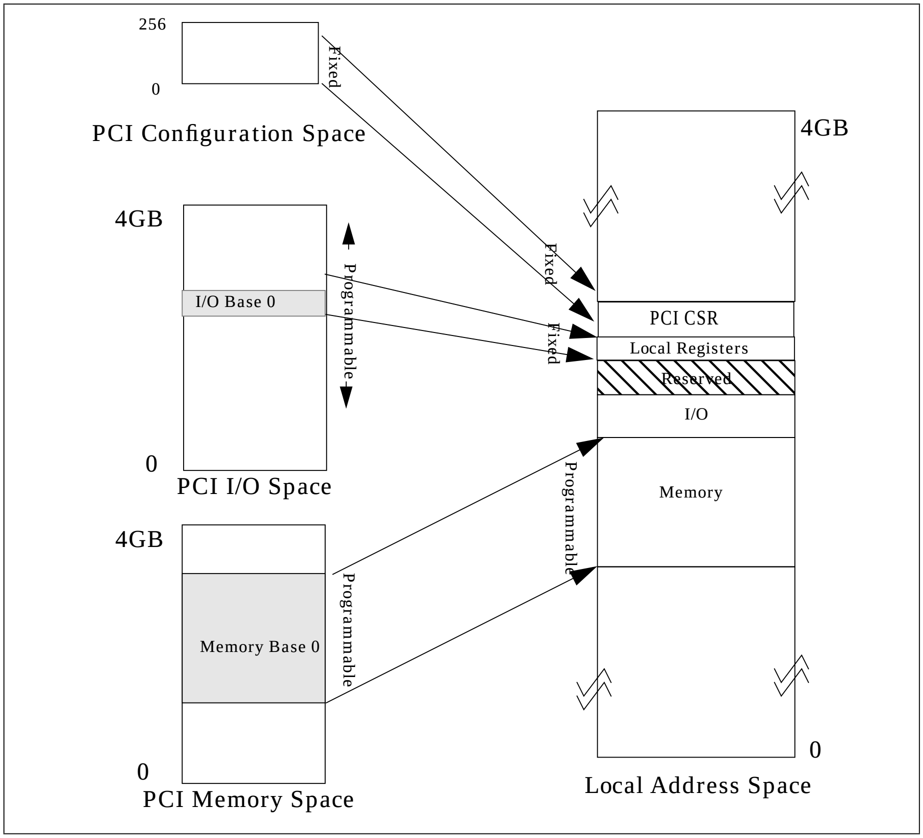


Figure 48: PCI target address space

Memory read/write

Devices on the PCI bus access local memory space when their transactions are within the range specified by the memory base address register PCI.MBAR[0][1] and the PCI to local address space window is enabled (PCI.LSR[0].ENABLE = 1). Otherwise, devices on the PCI bus cannot access local memory space. PCI.LSR[0]

1. Note that this implementation supports 1 MBAR. Subsequent implementations may support multiple MBARs.

allows the overall window for local memory space to be set. This is of power two size and can be aligned in a range from 64 Kbytes to 512 Mbytes.

For memory region access, PCI target transactions use a block memory sizing and locating scheme, with up to eight banks of local addresses being mapped. Each region in the local address space has a programmable memory aperture size to the power of 2, with a minimum of 64 Kbytes and a maximum of 512 Mbytes.

A region's memory space is accessed when a PCI bus transaction is within the range specified by the region base address register PCI.RBAR[N] and the region address space window is enabled (PCI.RSR[N].ENABLE = 1). Read accesses which do not map into an enabled region or region configuration register result in no access being passed to the SuperHyway and a PCI target abort signal being generated. Write accesses are accepted and ignored silently. Note that at reset none of the regions are enabled.

In addition two or more regions may have overlapping windows. When PCI accesses fall into more than one region the lowest numbered region accepts the access and the address is translated accordingly.

When performing an address translation the lower 16 bits of a PCI access are mapped unchanged to the local address. The upper 16 bits are translated into the upper part of the local memory address using the region space register PCI.RSR[N] and the region address register PCI.RLAR[N].

The upper 16 bits in PCI.RSR[0] specify the bits in the PCI.MBAR[0] register to be used. If a bit equals 0 in PCI.RSR[0] the corresponding bit in PCI.MBAR[0] is compared with the PCI address. The bits in the PCI.RLAR[N] register replace those of the PCI address to form the local address when there is a match. When a bit equals 1 in **PCI.RSR[n],** no comparison is made and the corresponding bit from the PCI address is taken.

The process is illustrated in *Figure 49*.

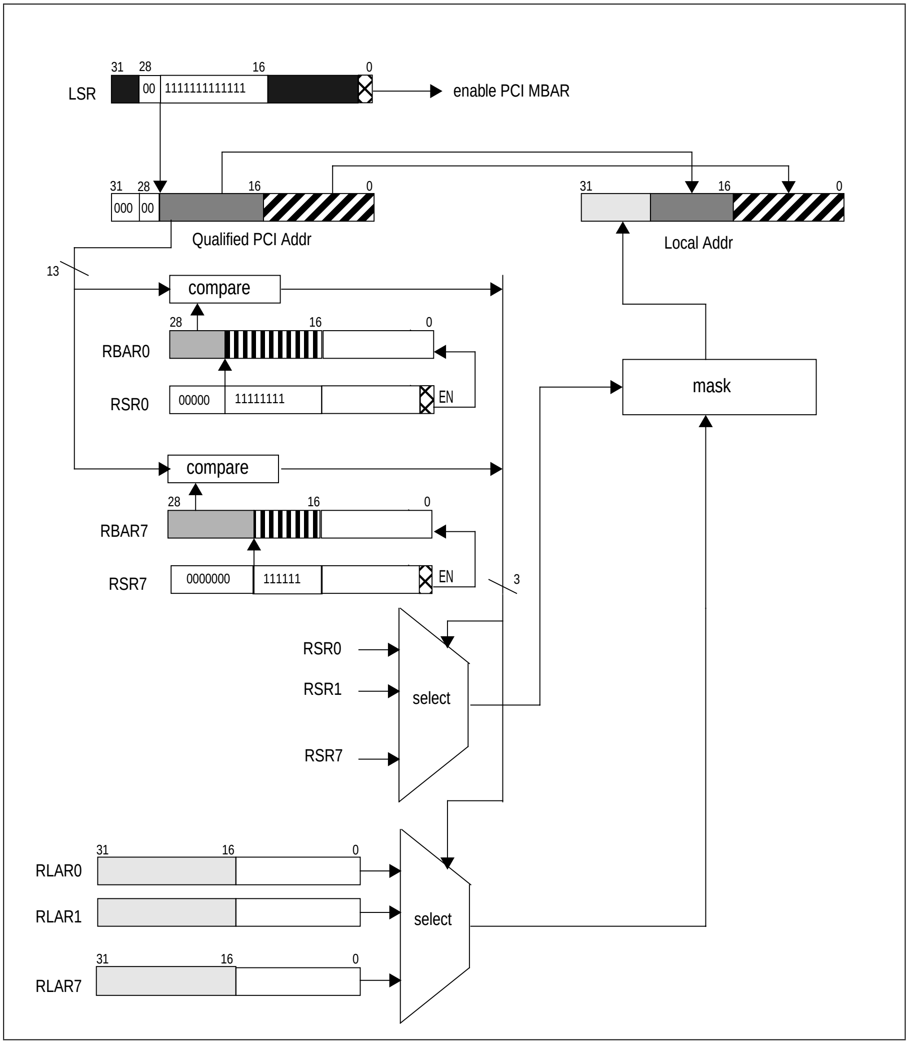


Figure 49: Address translation process

Regions can be located anywhere within the PCI target memory space (as defined by PCI.MBAR[0] and PCI.LSR[0]) and local address space so long as the base address is on a boundary which is an integer multiple of the region's size. Region configuration registers occupy a fixed window of 64 Kbytes on the PCI bus, by default at the base of target memory space. Using PCI.WCBAR the base address may be redefined to any multiple of 64 Kbytes up to the limit in the LSR.

Region configuration space may be hidden beneath a memory region so that PCI address space is used more economically. In this case, when an access is performed at a PCI address which maps to both a region and configuration space, the address translation is performed. If none of the memory regions are enabled then region configuration registers are visible and occupy 64 Kbytes on the PCI bus. The region configuration registers accept burst writes from PCI but not burst reads, which cause the PCI master to time-out.

We recommend using cache line wrap mode access (PCI bits AD[1:0] = 10) to PCI memory as a linear burst transaction could start within a memory mapped region, but continue beyond the upper boundary of that region. For more details about cache line wrap mode see the *PCI Local Bus Specification*.

I/O read/write

PCI I/O accesses are mapped to the PCI controller's register space using a 256-byte window. This window contains the local register bank as described in *Table 72 on page 185*.

A PCI I/O read/write occurs if the address of a transaction on the PCI bus is within the range specified by the I/O base address register PCI.IBAR (see *Table 113 on page 261*). If the upper 24 bits [31:8] of the PCI I/O address on the PCI bus match the contents of PCI.IBAR, then bits 7 to 2 of the address select a register to access.

This process is illustrated in *Figure 50*.

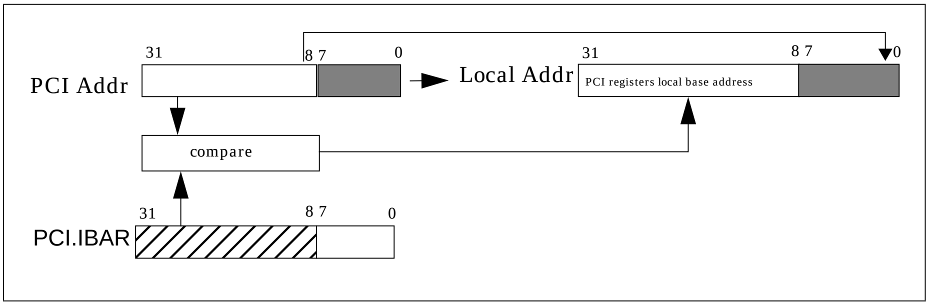


Figure 50: PCI I/O to local access

Configuration read/write

A PCI configuration read/write will access the CSR registers in the PCI controller when the PCI_IDSEL pin for the ST40 is asserted. Bits [7:2] in the PCI address field are used to select the register to read/write.

4.3.3 PCI errors

This section describes the error conditions and the behavior of the PCI module when it encounters them.

PCI target errors

The PCI module detects parity errors on address, read data and write data.

Address parity error

When PCI_PAR driven by the PCI master differs from the expected parity for the address and command the following process occurs.

1 If the address or command matches any of the base address registers then the cycle is claimed as normal and the transaction proceeds as normal.

2 The detected parity error flag PCI.STATUS.PCI_DPE (bit 15) is set.

3 If the parity error response flag PCI.CMD.PCI_PER (bit 6) is 1 and the NOT_PCI_SERR driver is enabled, (PCI.CMD.PCI_SERRE (bit 8) is 1), then NOT_PCI_SERR is asserted for one cycle and the signaled system error flag PCI.STATUS.PCI_SSE (bit 14) is set.

Read data parity error

A read data parity error is detected when the PCI master asserts NOT_PCI_PERR in response to read data driven by PCI module. No action is taken.

Write data parity error

A write data parity error is detected when the PCI_PAR that is received by the PCI module differs from the expected parity for data and byte enables, the following procedure occurs.

1 The detected parity error flag PCI.STATUS.PCI_DPE (bit 15) is set.

2 NOT_PCI_PERR is asserted if the command register parity error response flag PCI.CMD.PCI_PER (bit 6) is set to 1.

Errors as PCI master

In addition to parity errors on read data and write data, the module has to deal with the target aborting on read or write or the master itself aborting. These are described below.

Read data parity error

The PCI module detects a read data parity error when the PCI_PAR that is received by the PCI module differs from the expected parity for data and byte enables. The following procedure occurs:

1 The detected parity error flag PCI.STATUS.PCI_DPE (bit 15) is set.

2 If the parity error response flag PCI.CMD.PCI_PER (bit 6) is set, then the data parity error detected flag PCI.STATUS.PCI_MDPE (bit 8) is set and the PCI module asserts NOT_PCI_PERR.

3 A SuperHyway error response is returned to the local initiator.

Write data parity error

The PCI module detects a write data parity error when the PCI target asserts NOT_PCI_PERR in response to write data driven by the PCI module. The following procedure occurs:

1 If the parity error response flag PCI.CMD.PCI_PER (bit 6) is set, then the data parity error detected flag PCI.STATUS.PCI_MDPE (bit 8) is set.

2 A SuperHyway error response is returned to the local initiator.

Target abort on read

When the PCI target signals a target abort, a SuperHyway error response is returned to the local initiator.

Target abort on write

When the PCI target signals a target abort, the following procedure occurs:

1 The received target abort flag PCI.STATUS.PCI_RTA (bit 12) is set.

2 A SuperHyway error response is returned to the local initiator.

Master abort

A master abort occurs when NOT_PCI_DEVSEL is not asserted within five cycles after the PCI module asserts NOT_PCI_FRAME. This causes the following procedure to occur:

1 The received master abort flag PCI.STATUS.PCI.RMA (bit 13) is set.

2 A SuperHyway error response is returned to the local initiator.

4.4 Operation

4.4.1 Operation mode

The PCI_HOST strapping pin is sampled at reset and determines the operating mode of the PCI controller.

Function	Host bridge (PCI_HOST = 1)	Satellite (PCI_HOST = 0)
PCI arbitration	Provides arbitration	Bus requester
PCI interrupts	Interrupt controller Receives interrupts on NOT_PCI_INTA	Generates interrupt on NOT_PCI_INTA
Configuration registers	Generates configuration cycles to external PCI devices Internal configuration registers also accessed through configuration cycles.	Accepts configuration cycles from external host bridge
NOT_PCI_IDSEL	Unused	Used when target for configuration cycles

Table 69: Operating mode

4.4.2 PCI bus arbitration

In satellite mode PCI bus arbitration is performed by an external agent.

In host mode the PCI controller provides arbitration of the PCI bus. Support is provided for up to 4 external masters in addition to the ST40.

The PCI bus arbiter supports two modes to determine the priority of devices. Fixed priority and pseudo round robin.

Fixed priority arbitration (PCI.CR.PCI_BMAM = 0)

The priority of devices is fixed at the following default value.

PCI controller > Device 0 > Device 1 > Device 2 > Device 3

The ST40 PCI controller always takes priority.

Pseudo round robin arbitration (PCI.CR.PCI_BMAM = 1)

In pseudo round robin mode the priority of devices at any time is ordered by when each device was last granted. The least-recently granted device has the highest priority and the most-recently granted device has the lowest priority. The initial priority is the same as for fixed priority mode.

For example, if after reset device 1 is granted the bus and completes a data transfer the priority is set at the following value.

PCI controller > Device 0 > Device 2 > Device 3 > Device 1

Then after the PCI controller has completed a PCI bus transaction the priority is changed to the following value.

Device 0 > Device 2 > Device 3 > Device 1 > PCI controller

Then after device 3 has completed a PCI bus transaction the priority is changed to the following value.

Device 0 > Device 2 > Device 1 > PCI controller > Device 3

4.4.3 Configuration access

The PCI module supports configuration mechanism 1 support as described in the PCI specification. The PCI PIO address register (PCI.PAR) and the PCI PIO data register (PCI.PDR), in the terminology of the PCI specification, correspond to the configuration address register and the configuration data register respectively.

First set PCI.PAR then a read or write to the PCI.PDR register causes a configuration cycle to be issued on PCI bus.

There are two types of configuration transfers: type 1 and type 0. These transactions are differentiated by the values on PCI_AD[1:0]. A type 0 configuration transaction (when PCI_AD[1:0] = 00) is used to select a device on the bus where the transaction is being run (bus number is 0). A type 1 configuration transaction (when PCI_AD[1:0] = 01) is used to pass a configuration request to another bus segment.

In type 1 configuration transfers PCI.PAR is PCI_AD[31:2] in address phase of configuration access (bits [1:0] = 1'b10)

In type 0 transfers, bit [10:2] is passed on to PCI bus unchanged, but PCI_AD[31:11] is changed so that they may be used as IDSEL signals. In general setting device number to n = [0:15] makes PCI_AD[31 - N] = 1 and other PCI_AD = 0, for example setting:

- device number = 0 makes PCI_AD[31] = 1 and others=0,
- device number = 1 makes PCI_AD[30] = 1 and others=0,
- device number = 15 makes PCI_AD[16] = 1 and others=0.

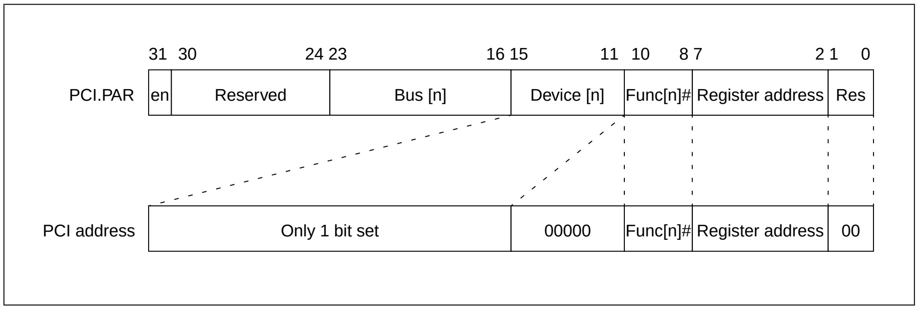


Figure 51: Address generate for type 0 configuration

When making a configuration access, A PCI master abort (no device connected) will not cause an interrupt. In this case, configuration writes end normally and configuration reads return a value of 0.

4.4.4 PCI interrupts

The PCI module is capable of generating the following types of interrupts.

Interrupt name	Causes
PCI_SERR_INT	Detection that the NOT_PCI_SERR pin is asserted (in host mode).
PCI_ERR_INT	PCI errors as reported in the PCI.INT register PCI arbiter error as reported in the PCI.AINT register All these interrupt causes may be masked by using the PCI.INTM and PCI.AINTM registers respectively.
PCI_AD_INT	Detection that the NOT_PCI_INTA is asserted (in host mode)
PCI_PWR_DWN	Detection that the PCI.CSR register has been updated to cause a transition to the D0 (normal operation) state (in satellite mode). The PCI_PMD0 bit of the PCI.PINT register is set. This interrupt may be masked by the PCI.PINTM register.
	Detection that the PCI.CSR register has been updated to cause a transition to the D3 (low power) state (in satellite mode).The PCI_PMD3H bit of the PCI.PINT register will be set. This interrupt may be masked by the PCI.PINTM register

Table 70: PCI interrupt types

4.4.5 Reset

Power-on reset when host

When the PCI module is configured as the PCI host, PCI_RST_OUT_N is asserted whenever the ST40 is powered on.

PCI_CFINT should be asserted at the same time as reset is deasserted.

Reset when satellite

Some implementations may implement a NOT_PCI_RST input pin allowing the PCI bus host to reset the ST40 PCI controller.

Other implementations may not implement a pci reset input pin. In such circumstances the NOT_PCI_RST signal may be connected to the ST40 global reset taking care to observe the timing constraints involved. See the datasheet for more details.

When the PCI bridge is initializing, the bridge returns RETRY in response to **read** commands. When the ST40 host processor has been internally configured, PCI_CFINT is set to 1.

Software controlled reset of the PCI bus

The PCI_RST_OUT_N pin may be controlled by writing to PCI.CR.PCI_RSTCTL. During hard reset of the ST40, the NOT_PCI_RST pin is asserted causing the PCI bus to be reset. The NOT_PCI_RST pin will remain asserted until software writes to PCI.CR.PCI_RSTCTL to de-assert the pin.

Software reset of the PCI controller

The ST40’s PCI controller may be reset by writing to PCI.CR.PCI_SOFT_RESET. This causes all states within the PCI controller to be reset to its power-on reset values. Note that any outstanding transactions the PCI controller may have with any other ST40 module (for example the CPU) are responded to before the reset completes.

4.4.6 Clocking

The PCI controller has two clock domains. 1, containing the local register bank and customized logic, is clocked synchronously to the ST40’s bus clock. The other, which contains the CSR registers and core PCI logic, is synchronous to the PCI bus. This second clock domain is either generated from a dedicated on-chip PLL or input from the PCI bus.

Controlling the state and frequencies of the on-chip clock domains is described in the clock chapter. The selection of the PCI bus clock may be observed from software by setting PCI.CR.PCI.HOSTNS and PCI.CR.PCI_CLKENS.

4.4.7 Power management

The PCI interface supports PCI power management (version 1.0 compatible) configuration registers. This comprises:

- support for the PCI power management control configuration registers,
- support for the power-down/restore request interrupts from hosts on the PCI bus.

The two main configuration registers for PCI power management control are:

- PCI.PMC,
- PCI.PMCSR.

The PCI.PMC register indicates that only D0 and D3 power states are supported and that the ST40 does not assert a NOT_PCI_PME pin.

PCI.PMCSR is used when the ST40’s PCI controller is in satellite mode. The PCI host can write to the PCI_PS field to effect a power state transition with the ST40 PCI controller. Any valid state change may cause an interrupt (dependent on the settings of the power management interrupt and mask registers). Software running on the ST40 then has the responsibility for taking action appropriate to the requested power state transition.

All internally generated clocks for the PCI controller may be stopped and hence put the controller into a low power state by the mechanisms described in the clock chapter. Note that when in host mode ST40 software has the responsibility for managing the power states of other devices on the PCI bus. When the ST40 PCI controller is in satellite mode, software has the responsibility for effecting power transitions which may be signaled using the PCI.PMCSR register by the host.

4.4.8 Endian issues

The PCI bus is little endian. The PCI module makes no allowances for local systems which may be operating in big endian mode. It is the responsibility of system software to make the appropriate adjustments in order to support particular applications.

4.5 Registers

The PCI controller's registers are grouped into two banks:

- PCI configuration space registers (CSR),
- PCI local registers.

CSR registers conform to those specified as the configuration header region in the PCI local bus specification 2.1. The local registers are specific to the ST40's implementation of the PCI bus interface.

The base addresses given are offsets from the PCI module control block. This value is given in the system address map.

Register bank	Offset	Accessible in	Register listing
Local (including VCR)	0x0000 0000	Local address space PCI memory space	*Section 4.5.4: Local register bank on page 192*
CSR	0x0001 0000	Local address space PCI configuration space PCI I/O space	*Section 4.5.7: CSR register bank on page 245*

Table 71: PCI register banks

The register overviews for each bank are listed in the next section.

4.5.1 Local register bank overview

Register name	Description	Type	Address offset	Access size
PCI.VCR.STATUS	Version control register status, see *Table 75 on page 192*	RW	0x00000	32
PCI.VCR.VERSION	Version control register version, see *Table 76 on page 194*	RO	0x00008	32
PCI.CR	PCI control register, see *Table 77 on page 195*	RW	0x00010	32
PCI.LSR[0]	PCI local space register 0, see *Table 78 on page 202*	RW	0x00014	32
Reserved			0x00018	-
PCI.LAR[0]	PCI local address register 0, see *Table 79 on page 204*	RW	0x0001C	32
Reserved			0x00020	-
PCI.INT	PCI interrupt register, see *Table 80 on page 205*	RW	0x00024	32
PCI.INTM	PCI interrupt mask register, see *Table 81 on page 210*	RW	0x00028	32
PCI.AIR	PCI error address information register, see *Table 82 on page 215*	RO	0x0002C	32
PCI.CIR	PCI error command information register, see *Table 83 on page 216*	RO	0x00030	32
Reserved			0x00034 to 0x0003C	-
PCI.AINT	PCI arbiter interrupt register, see *Table 84 on page 218*	RW	0x00040	32

Table 72: PCI local register bank

Register name	Description	Type	Address offset	Access size
PCI.AINTM	PCI arbiter interrupt mask register, see *Table 85 on page 221*	RW	0x00044	32
PCI.BMIR	PCI error information, register of bus master, see *Table 86 on page 224*	RO	0x00048	32
PCI.PAR	PCI PIO address register, see *Table 87 on page 226*	RW	0x0004C	32
PCI.MBR	PCI memory space bank register, see *Table 88 on page 229*	RW	0x00050	32
PCI.IOBR	PCI I/O space bank register, see *Table 89 on page 230*	RW	0x00054	32
PCI.PINT	PCI power management interrupt register, see *Table 90 on page 231*	RW	0x00058	32
PCI.PINTM	PCI power management interrupt mask register, see *Table 91 on page 232*	RW	0x0005C	32
Reserved			0x00060 to 0x0006C	-
PCI.MBMR	PCI memory space bank mask register, see *Table 92 on page 233*	RW	0x00070	32
PCI.IOBMR	PCI I/O space bank mask register, see *Table 93 on page 235*	RW	0x00074	32
PCI.PDR	PCI PIO data register, see *Table 94 on page 236*	RW	0x00078	32
Reserved			0x0007C to 0x0007F	-

Table 72: PCI local register bank

Register name	Description	Type	Address offset	Access size
PCI.PERF[N]	Performance registers These are implementation specific. See the datasheet for details.	RW	0x00080 to 0x0008C	-
Local configuration registers, see *Table 73* below			0x00090 to 0x0FFFF	-

Table 72: PCI local register bank

4.5.2 Local configuration register bank overview

The local configuration registers define the access to, and mapping of, windows in PCI target memory space.

Register name	Description	Type	Address offset	Access size
PCI.WCBAR	Local configuration registers base address, see *Table 95 on page 237*	RW	0x0007C	32
PCI.LOCCFG_UNLOCK	Local configuration registers access control, see *Table 96 on page 238*	RW	0x00034	32
PCI.RBAR0	Region 0 base address register, see *Table 97 on page 239*	RW	0x00100	32
PCI.RSR0	Region 0 space register, see *Table 98 on page 240*	RW	0x00104	32
PCI.RLAR0	Region 0 local address register, see *Table 100 on page 243*	RW	0x00108	32
Reserved			0x0010C	-
PCI.RBAR1	Region 1 base address register, see *Table 97 on page 239*	RW	0x00110	32
PCI.RSR1	Region 1 space register, see *Table 98 on page 240*	RW	0x00114	32

Table 73: PCI local configuration register bank

Register name	Description	Type	Address offset	Access size
PCI.RLAR1	Region 1 local address register, see *Table 100 on page 243*	RW	0x00118	32
Reserved			0x0011C	-
PCI.RBAR2	Region 2 base address register, see *Table 97 on page 239*	RW	0x00120	32
PCI.RSR2	Region 2 space register, see *Table 98 on page 240*	RW	0x00124	32
PCI.RLAR2	Region 2 local address register, see *Table 100 on page 243*	RW	0x00128	32
Reserved			0x0012C	-
PCI.RBAR3	Region 3 base address register, see *Table 97 on page 239*	RW	0x00130	32
PCI.RSR3	Region 3 space register, see *Table 98 on page 240*	RW	0x00134	32
PCI.RLAR3	Region 3 local address register, see *Table 100 on page 243*	RW	0x00138	32
Reserved			0x0013C	-
PCI.RBAR4	Region 4 base address register, see *Table 97 on page 239*	RW	0x00140	32
PCI.RSR4	Region 4 space register, see *Table 98 on page 240*	RW	0x00144	32
PCI.RLAR4	Region 4 local address register, see *Table 100 on page 243*	RW	0x00148	32
Reserved			0x0014C	-
PCI.RBAR5	Region 5 base address register, see *Table 97 on page 239*	RW	0x00150	32
PCI.RSR5	Region 5 space register, see *Table 98 on page 240*	RW	0x00154	32

Table 73: PCI local configuration register bank

Register name	Description	Type	Address offset	Access size
PCI.RLAR5	Region 5 local address register, see *Table 100 on page 243*	RW	0x00158	32
Reserved			0x0015C	-
PCI.RBAR6	Region 6 base address register, see *Table 97 on page 239*	RW	0x00160	32
PCI.RSR6	Region 6 space register, see *Table 98 on page 240*	RW	0x00164	32
PCI.RLAR6	Region 6 local address register, see *Table 100 on page 243*	RW	0x00168	32
Reserved			0x0016C	-
PCI.RBAR7	Region 7 base address register, see *Table 97 on page 239*	RW	0x00170	32
PCI.RSR7	Region 7 space register, see *Table 98 on page 240*	RW	0x00174	32
PCI.RLAR7	Region 7 local address register, see *Table 100 on page 243*	RW	0x00178	32
Reserved			0x0017C	-

Table 73: PCI local configuration register bank

4.5.3 Configuration space register (CSR) bank overview

All PCI functions possess a block of 64 configuration doublewords reserved for the implementation of its configuration registers. The format and usage of the first 16 doublewords is prescribed by the PCI specification. This area is referred to here as the CSR registers but in PCI literature in general this is also commonly referred to as the configuration header region (or simply header space).

[31:24]	[23:16]	[15:8]	[7:0]	CSR offset
DeviceID		Vendor ID		00
Status		Command		04
Class code			Revision	08
Bist	Header	Latency	Cacheline	0C
Memory base address				10
■	■	■	■	14
IO base address				18
■	■	■	■	1C
▒	▒	▒	▒	20
▒	▒	▒	▒	24
▒	▒	▒	▒	28
Subsystem ID		Vendor ID		2C
▒	▒	▒	▒	30
■	■	■	cap_ptr	34
■	■	■	■	38
max latency	min_gnt	interrupt pin	interrupt line	3C
■	■	retry timout	trdy time-out	40

Table 74: CSR bank layout

[31:24]	[23:16]	[15:8]	[7:0]	CSR offset
■	■	■	■	44 ... D8
Power management capabilities		Next ptr	Capability ID	DC
Data	PMCSR _BSE	Power management PMCSR		E0
■	■	■	■	E4 FF

□	Read/write
░	Read-only
■	Reserved
▒	Don’t care

Table 74: CSR bank layout

4.5.4 Local register bank

PCI.VCR.STATUS

This register defines information available to the system, specifically debug, to determine how this module has interacted with the system, and, if any erroneous requests have occurred during operation of that module.

This information is generally used to allow debug software to determine which modules in the system have caused the system to fail, and supply information about how that failure occurred.

PCI.VCR.STATUS				**0x0000**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PERROR	[0:7]	8	Yes	SuperHyway error	RW
	Operation		Each bit corresponds to an erroneous SuperHyway operation being detected. The meaning of each bit is as follows: 0: Error response received 1: Error response returned 2: Access to undefined location accepted 3: Unsolicited response received 4: Reserved, write 0, read undefined 5: Unsupported operation accepted		
	Read		Returns current value		
	Write		For bits [0:7]: 0: No action 1: Reset bit		
	Hard reset		0		

Table 75: PCI.VCR.STATUS

PCI.VCR.STATUS				0x0000	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MERROR	[8:15]	8	Yes	Module error	RW
	Operation		Each bit corresponds to an erroneous module operation being detected. 0: If set an alignment error has occurred An alignment error occurs if a channel is mis-programming with a unalignable transfer on a channel which enforces alignment checks. [7:1]: Reserved		
	Read		Returns current value		
	Write		For bits [8:15]: 0: No action 1: Reset bit		
	Hard reset		0x00		
MOD_ID	[16:31]	16	No	Module identity	RO
	Operation		Indicates the module type		
	Read		Returns module type 0x5043		
	Write		Ignored		
	Hard Reset		0x5043		

Table 75: PCI.VCR.STATUS

PCI.VCR.VERSION

This register defines the module type and revision number.

PCI.VCR.VERSION				0x0008	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MOD_VER	[0:15]	16	No	Module version	RO
	Operation		Indicates the module type		
	Read		Returns module type 0x0001		
	Write		Ignored		
	Hard Reset		0x0001		
MOD_SIZE	[16:31]	16	No	Module size	RO
	Operation		Indicates module size defined in 64k blocks		
	Read		Returns current value 0x07FF		
	Write		Ignored		
	Hard reset		0x07FF		

Table 76: PCI.VCR.VERSION

PCI control register

The PCI.CR is a 32-bit register which controls the PCI interface. This register is write-restricted to give some protection against erroneous programming. It is only possible to update this register when the value of the top byte (that is bits[31:24]) of the write data is 0x5A. Other values cause the write to be ignored.

PCI.CR				0x00010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
LOCK	[31:24]	8	-	Lock	RW
	Operation		Reserved		
	Read		0x00		
	Write		0x5A		
	Hard reset		0x00		
-	[23:13]	11	-	Reserved	RES
	Operation		Reserved		
	Read		13’b0000000000000		
	Write		-		
	Hard reset		13’b0000000000000		
PCI_SOFT_RESET	12	1	Yes	PCI soft reset control	RW
	Operation		Soft resets the PCI core		
	Read		0: Normal operating state 1: Soft reset in progress		
	Write		0: No effect 1: Soft reset the core Updates current value (PCI.CR.LOCK = 0x5A)		
	Hard reset		0		

Table 77: PCI.CR

PCI.CR				0x00010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PFCS	11	1	-	PCI pre-fetch command setting	RW
	Operation		1’b0: always 8 byte pre-fetching 1’b1: always 32 byte pre-fetching		
	Read		Returns current value		
	Write		Updates current value (PCI.CR.LOCK = 0x5A)		
	Hard reset		1’b0		
-	10	1	-	Reserved	RES
	Operation		Reserved		
	Read		13’b0000000000000		
	Write		-		
	Hard reset		13’b0000000000000		

Table 77: PCI.CR

PCI.CR				0x00010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PFE	9	1	-	PCI pre-fetch enable	RW

	Operation		1’b0: disable
			1’b1: enable
			When the ST40 is the target of a PCI command pre-fetch being enabled is interpreted as follows:
			IO command = Never pre-fetch, always 4-byte access
			Memory Read Multiple = Always pre-fetch in units of 32 bytes
			Memory Read Line = Pre-fetch single 32-byte packet
			Memory Read
			`PCI_PFE = 0`
			`only request 4 bytes`
			`wait for data to be returned`
			`wait for [n] cycles to check for phoenix STOP signal`
			`IF stop finish`
			`ELSE wait for [n] cycles and make another 4-byte read`
			`PCI_PFE = 1`
			`when PCI_PFCS = 1 treat as Memory Read Multiple`
			`when PCI_PFCS = 0 request 8 bytes`
			`request 8 bytes when first 8 has returned until STOP indicates that core has required data`		

	Read		Returns current value		
	Write		Updates current value (PCI.CR.LOCK = 0x5A)		
	Hard reset		1’b0		

Table 77: PCI.CR

PCI.CR				0x00010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[8:7]	2	-	Reserved	RES
	Operation		Reserved		
	Read		13’b0000000000000		
	Write		-		
	Hard reset		13’b0000000000000		
PCI_BMAM	6	1	-	PCI bus master arbitration mode control	RW
	Operation		0: Fixed mode (device0 > device1 > device2 > device3 > device4) 1: Round robin		
	Read		Returns current value		
	Write		Updates current value (PCI.CR.LOCK = 0x5A)		
	Hard reset		1’b0		
PCI_HOST	5	1	-	PCI host status	RO
	Operation		Indicates wether the ST40 is the host or satellite		
	Read		0: Satellite mode 1: Host mode		
	Write		-		
	Hard reset		Determined by the PCI_HOST pin		

Table 77: PCI.CR

PCI.CR				0x00010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_CLKEN	4	1	-	PCI clock	RO
	Operation		Indicates whether the PCI module is clocked by the on-chip PLL or takes its clock externally.		
	Read		0: Clocked by internal clock. 1: Clocked by external clock		
	Write		-		
	Hard reset		See datasheet Some implementations may sample a PCI clock select pin on reset. Other implementations may use the PCI_HOST pin taking host mode to imply internal clock and vice versa.		
PCI_SOCS	3	1	-	PCI NOT_PCI_SERR output control by software	RW
	Operation		Asserts NOT_PCI_SERR for one PCI_CLK cycle Only when PCI.CMD.PCI_SERRE = 1 is 1’b1 effective.		
	Read		1’b0		
	Write		1’b0: NOT_PCI_RST high level output (negated) 1’b1: NOT_PCI_SERR asserted for one PCI_CLK cycle (normal mode) This bit uses normal mode only(PCI.CR.LOCK = 0x5A)		
	Hard reset		1’b0		

Table 77: PCI.CR

PCI.CR				0x00010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_IOCS	2	1	-	PCI NOT_PCI_INTA output control by software	RW
	Operation		This bit uses normal mode only		
	Read		Returns current value		
	Write		1'b0: NOT_PCI_INTA high level output		
			1'b1: NOT_PCI_INTA low level output		
			Updates current value (PCI.CR.LOCK = 0x5A)		
	Hard reset		1'b0		
PCI_RSTCTL	1	1	-	PCI NOT_PCI_RST output control by software	RW
	Operation		Controls the state of the NOT_PCI_RST pin		
	Read		Returns current value		
	Write		1'b1: NOT_PCI_RST low level output (asserted)		
			Following hard reset software should clear this bit to take the PCI bus out of reset.		
			Updates current value (PCI.CR.LOCK = 0x5A)		
	Hard reset		1'b1		

Table 77: PCI.CR

PCI.CR				0x00010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_CFINT	0	1	-	PCI internal register initialize control	RW
	Operation		PCI_CFINT is connected to an internal signal APP_TARGET_READY, which controls the target response to external PCI commands. When APP_TARGET_READY is 0, the bridge returns RETRY in response to **read** commands. This allows a period to set up the application side before an external PCI host reads information from the device to configure it. For example, the size of memory space needs to be set before an external PCI host allocates a position in the memory map. In host mode, the signal is redundant, since PCI configuration is done by the ST40. PCI_CFINT should be asserted at the same time as PCI reset is deasserted.		
	Read		Returns current value		
	Write		1’b0: PCI bridge initializing 1’b1: PCI bridge active Updates current value (PCI.CR.LOCK = 0x5A)		
	Hard reset		1’b0		

Table 77: PCI.CR

PCI local space register 0

The use of this register is described in *Memory read/write on page 172*.

PCI.LSR[0]				0x00014	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
ENABLE	0	1	No	Memory base address register 0 enable	RW
	Operation		0: PCI.MBAR[0] disabled 1: PCI.MBAR[0] enabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
-	[1:15]	15	No	Reserved	RO
	Operation		-		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 78: PCI.LSR[0]

PCI.LSR[0]				0x00014	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MASK	[16:28]	13	No	Base address mask	RW
	Operation		For each bit of this field: 0: the corresponding bit in PCI.MBAR[0] acts as a read/write bit 1: the corresponding bit in PCI.MBAR[0] acts as a read-only 0 bit The values of the PCI.LSR[0] register are constrained to those listed below. All other values are illegal. 0x0000 0001= 64 Kbytes 0x0001 0001: 128 Kbytes 0x0003 0001: 256 Kbytes 0x0007 0001: 512 Kbytes 0x000F 0001: 1 Mbyte 0x001F 0001: 2 Mbytes 0x003F 0001: 4 Mbytes 0x007F 0001: 8 Mbytes 0x00FF 0001: 16 Mbytes 0x01FF 0001: 32 Mbytes 0x03FF 0001: 64 Mbytes 0x07FF 0001: 128 Mbytes 0x0FFF 0001: 256 Mbytes 0x1FFF 0001: 512 Mbytes		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
-	[29:31]	3	No	Reserved	RO
	Operation		-		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 78: PCI.LSR[0]

PCI local address register 0

This register is no longer used. See *Memory read/write on page 172* for information about PCI target memory transactions.

PCI.LAR[0]				0x0001C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[0:15]	16	-	Reserved	RO
	Operation		-		
	Read		0		
	Write		Ignored		
	Hard reset		0x0000		
-	[16:31]	16	-	Reserved	RW
	Operation		Undefined		
	Read		Undefined		
	Write		Ignored		
	Hard reset		0x0000		

Table 79: PCI.LAR[0]

PCI interrupt register

This register records the occurrence of conditions which may cause a PCI interrupt.

When multiple interrupt conditions occur, only the first cause is registered.

When an interrupt type is disabled the cause is still registered in the corresponding bit, but no interrupt occurs

Note that some implementations may not provide all of the interrupt causes described in the PCI.INT register.

PCI.INT				0x00024	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:16]	16	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_MNLTDIM	15	1	-	PCI master non-lock transfer detection interrupt	RW
	Operation		PCI core has detected a PCI master non-lock transfer		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_TTADI	14	1	-	PCI target abort detection interrupt	RW
	Operation		PCI bridge detected illegal byte enable with I/O transfer at the time of target (target only)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		

Table 80: PCI.INT

PCI.INT				0x00024	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[13:10]	4	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_TMTO	9	1	-	PCI target memory read/write time-out interrupt	RW
	Operation		PCI bridge did not do retry processing into a specified clock at the time of the target. Memory transfer only (target)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_MDEI	8	1	-	PCI master function disable error interrupt	RW
	Operation		PCI bridge did master actuation (PIO transfer and PCI transfer) when PCI.CMD.PCI_BM = 1’b0 (master)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		

Table 80: PCI.INT

PCI.INT				0x00024	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_APEDI	7	1	-	PCI address parity error detection interrupt	RW
	Operation		PCI bridge detected address parity error (target only)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_SDI	6	1	-	NOT_PCI_SERR detection interrupt	RW
	Operation		PCI bridge detects NOT_PCI_SERR is asserted (master and target) PCI bridge in host mode only		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_DPEITW	5	1	-	PCI data parity error detection interrupt for target write	RW
	Operation		When the PCI bridge received target write transfer, a data parity error was detected. (when PCI.COM.PCI_PER = 1’b1) (target only)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		

Table 80: PCI.INT

PCI.INT				0x00024	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PEDITR	4	1	-	NOT_PCI_PERR detection interrupt for target read	RW
	Operation		When the PCI bridge received target read transfer, NOT_PCI_PERR signal was detected (target only)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_TADIM	3	1	-	PCI target abort detection interrupt for master	RW
	Operation		PCI bridge detected target abort (that is NOT_PCI_DEVSEL was negated prematurely) (master only) PCI bridge as master only		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_MADIM	2	1	-	PCI master-abort detection interrupt for master	RW
	Operation		PCI bridge detected master-abort (NOT_PCI_DEVSEL undetected) (master only)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		

Table 80: PCI.INT

PCI.INT				0x00024	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_MWPDI	1	1	-	PCI master write NOT_PCI_PERR detection interrupt	RW
	Operation		PCI bridge received NOT_PCI_PERR from the target at the time of the data write to the target (master only)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_MRDPEI	0	1	-	PCI master read data parity error detection interrupt	RW
	Operation		PCI bridge detected data parity error at the time of data read to target (master only)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		

Table 80: PCI.INT

PCI interrupt register mask

This register is the mask register for PCI.INT.

Note that some implementations may not provide masking control on all of the interrupt causes described in the PCI.Int register.

PCI.INTM				**0x00028**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:16]	16	-	Reserved	RW
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_MNLTDIM	15	1	-	PCI master non-lock transfer detection interrupt mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_MNLTDIM 1'b0: Inhibition PCI.INT.PCI_MNLTDIM		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		
PCI_TTADIM	14	1	-	PCI target abort detection interrupt mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_TTADIM 1'b0: Inhibition PCI.INT.PCI_TTADIM		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		

Table 81: PCI.INTM

PCI.INTM				0x00028	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[13:10]	4	-	Reserved	RW
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_TMTOM	9	1	-	PCI target memory read/write time out interrupt mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_TMTO 1'b0: Inhibition PCI.INT.PCI_TMTO		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		
PCI_MDEIM	8	1	-	PCI master function disable error interrupt mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_MDEI 1'b0: Inhibition PCI.INT.PCI_MDEI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		

Table 81: PCI.INTM

PCI.INTM				0x00028	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_APEDIM	7	1	-	PCI address parity error detection interrupt mask	RW
	Operation		1’b1: Permission PCI.INT.PCI_APEDI 1’b0: Inhibition PCI.INT.PCI_APEDI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		
PCI_SDIM	6	1	-	PCI NOT_PCI_SERR detection interrupt mask	RW
	Operation		1’b1: Permission PCI.INT.PCI_SDI 1’b0: Inhibition PCI.INT.PCI_SDI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		
PCI_DPEITWM	5	1	-	Data parity error detection interrupt for target write mask	RW
	Operation		1’b1: Permission PCI.INT.PCI_DPEITW 1’b0: Inhibition PCI.INT.PCI_DPEITW		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		

Table 81: PCI.INTM

PCI.INTM				**0x00028**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PEDITRM	4	1	-	NOT_PCI_PERR detection interrupt for target read mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_PEDITR 1'b0: Inhibition PCI.INT.PCI_PEDITR		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		
PCI_TADIMM	3	1	-	target abort detection interrupt for master mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_TADIM 1'b0: Inhibition PCI.INT.PCI_TADIM		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		
PCI_MADIMM	2	1	-	PCI master-abort detection interrupt for master mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_MADIM 1'b0: Inhibition PCI.INT.PCI_MADIM		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		

Table 81: PCI.INTM

PCI.INTM				0x00028	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_MWPDIM	1	1	-	PCI master write NOT_PCI_PERR detection interrupt mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_MWPDI 1'b0: Inhibition PCI.INT.PCI_MWPDI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		
PCI_MRDPEIM	0	1	-	PCI master read data parity error detection interrupt mask	RW
	Operation		1'b1: Permission PCI.INT.PCI_MRDPEI 1'b0: Inhibition PCI.INT.PCI_MRDPEI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		

Table 81: PCI.INTM

PCI error address information register

This register records PCI access address information at the time an interrupt condition is detected.

PCI.AIR				0x0002C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_AIR	[31:0]	32	-	PCI error address information register	RO
	Operation		Holds address information when PCI bridge indicates an error		
	Read		Returns current value		
	Write		-		
	Hard reset		32'hxxxxxxxx		

Table 82: PCI.AIR

Note: There may be circumstances when the address recorded in the PCI.AIR *register doesn't correspond to the error, for example when several errors occur close together.*

PCI error command information register

This register records PCI command information at the time an interrupt condition is detected.

PCI.CIR				0x00030	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PIOTEM	31	1	-	PCI PIO transfer error for master	RO
	Operation		Error producing at the time of PIO transfer (master)		
	Read		Returns current value		
	Write		-		
	Hard reset		1'bx		
-	[30:27]	4	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_RWTET	26	1	-	PCI read/write transfer error for target	RO
	Operation		Error producing at the time of read/write transfer (target)		
	Read		Returns current value		
	Write		-		
	Hard reset		1'bx		
-	[25:4]	22	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		

Table 83: PCI.CIR

PCI.CIR				0x00030	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_ECR	[3:0]	4	-	PCI error command register	RO
	Operation		Holds command information, when PCI bridge find an error		
	Read		Returns current value		
	Write		-		
	Hard reset		4’bxxxx		

Table 83: PCI.CIR

PCI arbiter interrupt register

In host mode, this register records the cause of an arbiter interrupt.

When multiple interrupts occur only the first cause is registered.

When interrupt is disabled, the cause is registered in the corresponding bit and no interrupt occurs.

Note: *Some implementations may not record all of the interrupt causes described in the* PCI.AINT *register.*

PCI.AINT				**0x00040**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:14]	18	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_MBI	13	1	-	PCI master-broken interrupt	RW
	Operation		NOT_PCI_FRAME not asserted within 16 clocks, although PCI bridge gave the bus		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		

Table 84: PCI.AINT

PCI.AINT				0x00040	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_TBTOI	12	1	-	PCI target bus time-out interrupt	RW
	Operation		NOT_PCI_TRDY or NOT_PCI_STOP have not asserted within 16 clocks (first data transfer) NOT_PCI_TRDY or NOT_PCI_STOP have not asserted within eight clocks (subsequent data transfers)		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_MBTOI	11	1	-	PCI master bus time-out interrupt	RW
	Operation		NOT_PCI_IRDY not returned within eight clocks		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
-	[10:4]	7	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_TAI	3	1	-	PCI target abort interrupt	RW
	Operation		NOT_PCI_DEVSEL negated during the transfer, when device other than the PCI bridge has bus		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		

Table 84: PCI.AINT

PCI.AINT				0x00040	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_MAI	2	1	-	PCI master-abort interrupt	RW
	Operation		Master abort detected when device other than the PCI bridge has bus		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_RDPEI	1	1	-	PCI read data parity error interrupt	RW
	Operation		Parity error detected at data read when device other than PCI bridge has bus.		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		
PCI_WDPEI	0	1	-	PCI write data parity error interrupt	RW
	Operation		Parity error detected at data write when device other than PCI bridge has bus		
	Read		Returns current value		
	Write		1’b1: Clear this register		
	Hard reset		1’b0		

Table 84: PCI.AINT

PCI arbiter interrupt mask register

This register is the mask register for PCI.AINT.

Note: *Some implementations may not record all of the interrupt causes described in the* PCI.AINT *register and therefore the mask in the* PCI.AINTM *will be ineffective.*

PCI.AINTM				0x00044	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:14]	18	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_MBIM	13	1	-	PCI master-broken interrupt mask	RW
	Operation		1’b1: Permission PCI.AINT.PCI_MBI 1’b0: Inhibition PCI.AINT.PCI_MBI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		
PCI_TBTOIM	12	1	-	PCI target bus time-out interrupt mask	RW
	Operation		1’b1: Permission PCI.AINT.PCI_TBTOI 1’b0: Inhibition PCI.AINT.PCI_TBTOI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		

Table 85: PCI.AINTM

PCI.AINTM				0x00044	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_MBTOIM	11	1	-	PCI master bus time-out interrupt mask	RW
	Operation		1’b1: Permission PCI.AINT.PCI_MBTOI 1’b0: Inhibition PCI.AINT.PCI_MBTOI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		
-	[10:4]	7	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_TAIM	3	1	-	PCI target abort interrupt mask	RW
	Operation		1’b1: Permission PCI.AINT.PCI_TAI 1’b0: Inhibition PCI.AINT.PCI_TAI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		
PCI_MAIM	2	1	-	PCI master-abort interrupt mask	RW
	Operation		1’b1: Permission PCI.AINT.PCI_MAI 1’b0: Inhibition PCI.AINT.PCI_MAI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		

Table 85: PCI.AINTM

PCI.AINTM				0x00044	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_RDPEIM	1	1	-	PCI read data parity error interrupt mask	RW
	Operation		1'b1: Permission PCI.AINT.PCI_RDPEI 1'b0: Inhibition PCI.AINT.PCI_RDPEI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		
PCI_WDPEIM	0	1	-	PCI write data parity error interrupt mask	RW
	Operation		1'b1: Permission PCI.AINT.PCI_WDPEI 1'b0: Inhibition PCI.AINT.PCI_WDPEI		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1'b0		

Table 85: PCI.AINTM

PCI arbiter bus master information register

In host bridge mode, this register records bus master when the interrupt is invoked by PCI.AINT.

When multiple interrupts occur only the first cause is registered.

When interrupt is disabled the cause is registered in the corresponding bit and no interrupt occurs

Note: *Some implementations may not record the information described in the* PCI.BMIR.REGISTER.

PCI.BMIR				0x00048	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:5]	27	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_REQ4BME	4	1	-	PCI REQ4 bus master error	RO
	Operation		-		
	Read		Returns current value		
	Write		-		
	Hard reset		1'bx		
PCI_REQ3BME	3	1	-	PCI REQ3 bus master error	RO
	Operation		-		
	Read		Returns current value		
	Write		-		
	Hard reset		1'bx		

Table 86: PCI.BMIR

PCI.BMIR				0x00048	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_REQ2BME	2	1	-	PCI REQ2 bus master error	RO
	Operation		-		
	Read		Returns current value		
	Write		-		
	Hard reset		1’bx		
PCI_REQ1BME	1	1	-	PCI REQ1 bus master error	RO
	Operation		-		
	Read		Returns current value		
	Write		-		
	Hard reset		1’bx		
PCI_REQ0BME	0	1	-	PCI REQ0 bus master error	RO
	Operation		SH-4/ST40 PCI bridge occurred error		
	Read		Returns current value		
	Write		-		
	Hard reset		1’bx		

Table 86: PCI.BMIR

PCI PIO address register

Refer to *Section : Configuration read/write on page 171* for how to make configuration space accesses using this register.

PCI.PAR				0x00004C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_CCIE	31	1	-	PCI configuration cycle issue enable	RO
	Operation		Hard fixed		
	Read		1'b1		
	Write		-		
	Hard reset		1'b1		
-	[30:24]	7	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_BN	[23:16]	8	-	PCI bus number	RW
	Operation		8'h00: Bus number 0 (to the device on the bus that this bridge is connected the issue of configuration cycle) 8'h01: Bus number 1 to 8'hff = Bus number 255		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0x00		

Table 87: PCI.PAR

PCI.PAR				0x00004C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_DN	[15:11]	5	-	Device number	RW

	Operation		Shows the device number
			One of bits 31 to 16 of the PCI_AD bus which corresponds to the device number is driven to 1. Below shows the relationship between the device number and PCI_AD[31:16]. When the device number is 16 or greater PCI_AD[31:16] is all 0's.
			`for all pci_dn ≤ 15`
			`pci_ad[31 - pci_dn] = 1`
			`else pci_ad = 0`
			`for all pci_dn ≥16`
			`pci_ad [31, 16] = 0`
			For example:
			if PCI_DN = 0, PCI_AD[31] = 1 and PCI_AD[16:30] = 0,
			if PCI_DN = 7, PCI_AD[24] = 1, PCI_AD[16:23] and PCI_AD[25:31] = 0,
			PCI_DN =15, PCI_AD[16] = 1 and PCI_AD[17:31] = 0.		

	Read		Returns current value		
	Write		Updates current value		
	Hard reset		5'bxxxxx		
PCI_FN	[10:8]	3	-	Function number	RW
	Operation		3'b000: Single function device or function 0 of multi-function device 3'b001: Function 1 of multi-function device to 3'b111: Function 7 of multi-function device		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		3'bxxx		

Table 87: PCI.PAR

PCI.PAR				0x00004C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_CRA	[7:2]	6	-	Configuration register address	RW
	Operation		Long word boundary		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		6’bxxxxxx		
PCI_TYPE	[1:0]	2	-	Configuration access type	RW
	Operation		Specifies whether this access is a type 1 or a type 0 configuration access		
	Read		Returns current value		
	Write		00: Type 0 access 01: Type 1 access Other values are not allowed		
	Hard reset		2’b00		

Table 87: PCI.PAR

PCI memory space bank register

This register holds the upper bits of the PCI memory address space used when translating from the local address space. It is used in conjunction with the PCI.MBMR register described in *PCI memory space bank mask register on page 233*.

Refer to *Memory read/write on page 168* for details of how to access PCI Memory space.

PCI.MBR				**0x00050**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PMSBA	[31:16]	16	-	Memory space bank address register	RW
	Operation		-		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0x00		
-	[15:0]	16	-	Reserved	RES
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		

Table 88: PCI.MBR

PCI I/O space bank register

Refer to *Section : I/O read/write on page 169* for details on how to access PCI I/O space.

PCI.IOBR				0x00054	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PIOSBA	[31:16]	16	-	I/O space bank address register	RW
	Operation		-		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		16’bxxxx xxxx xxxx xx		
-	[15:0]	16	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		

Table 89: PCI.IOBR

PCI power management interrupt register

This register controls the power management interrupts.

PCI.PINT				0x00058	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PMD0	0	1	-	PCI Power Management D0 status transition Interrupt	RW
	Operation		Transition request to D0 (hot) normal operating power state		
	Read		Returns current value		
	Write		1’b1: Clear this bit		
	Hard reset		1’b0		
PCI_PMD3H	1	1	-	Power management d3hot status transition interrupt	RW
	Operation		Transition request to D3 (hot) low power state		
	Read		Returns current value		
	Write		1’b1: Clear this bit		
	Hard reset		1’b0		
-	[2:31]	30	-	Reserved	RES
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		

Table 90: PCI.PINT

PCI power management interrupt mask register

This register is the mask register for PCI.PINT.

PCI.PINTM				0x0005C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PMD0M	0	1	-	Power management d0 status transition interrupt mask	RW
	Operation		1: PCI.PINT.PCI_PMD0 inhibited 0: PCI.PINT.PCI_PMD0 allowed		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		
PCI_PMD3HM	1	1	-	Power management d3 hot status transition interrupt mask	RW
	Operation		0: PCI.PINT.PCI_PMD3H inhibited 1: PCI.PINT.PCI_PMD3H allowed		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		
-	[2:31]	30	-	Reserved	RES
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		

Table 91: PCI.PINTM

PCI memory space bank mask register

This register is the mask register for PCI.MBR (see *PCI memory space bank register on page 229*). Refer to *Memory read / write on page 168* on accessing PCI memory space

PCI.MBMR				0x00070	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:27]	5	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
MR	[26:16]	11	-	Memory space bank address mask register	RW
	Operation		Specifies the mask for local to PCI memory address translation and the PCI memory address range. There are 12 aperture sizes represented by: $2^n - 1$ (n = 0 to 11) => PCI aperture size of $2^{(n + 16)}$ These are: 000 0000 0000: 64 Kbytes 000 0000 0001: 128 Kbytes 000 0000 0011: 256 Kbytes to111 1111 1111: 128 Mbytes If the aperture size is less than the PCI memory space then the address mapping wraps around so that the PCI memory address may be mapped by multiple local addresses. If the aperture size is greater than the PCI memory space then the aperture is truncated to the memory space size. See *Section 4.2: Local address map on page 165*		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		11’b000 0000 0000		

Table 92: PCI.MBMR

PCI.MBMR				0x00070	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[15:0]	21	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		

Table 92: PCI.MBMR

PCI I/O space bank mask register

This register is the mask register for PCI.IOBR.

Refer to *Section : I/O read/write on page 169* on accessing PCI I/O space

PCI.IOBMR				0x00074	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:24]	8	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_IOBAMR	[23:16]	8	-	I/O space bank address mask register	RW
	Operation		Specifies the mask for local to PCI I/O address translation and the PCI I/O address range. There are nine aperture sizes represented by: 2^n - 1 (n = 0 to 8) => PCI I/O aperture size of $2^{(n + 16)}$ These are: 0000 0000: 64 Kbytes 0000 0000: 128 Kbytes 0000 0011: 256 Kbytes 0000 0111: 512 Kbytes 0000 1111: 1 Mbyte up to 1111 1111: 16 Mbytes If the aperture size is less than the PCI IO space then the address mapping wraps around so that the PCI IO address may be mapped by multiple local addresses. If the aperture size is greater than the PCI IO space then the aperture is truncated to the IO space size. See *Section 4.2: Local address map on page 165*		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		5'bxxxx x		

Table 93: PCI.IOBMR

PCI.IOBMR				0x00074	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[15:0]	16	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		

Table 93: PCI.IOBMR

PCI PIO data register

Accessing this register generates a configuration cycle on the PCI bus. See *Section : Configuration read / write on page 171* for details.

PCI.PDR				0x00078	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PDR	[31:0]	32	-	PIO data register	RW
	Operation		Reading and writing generates the configuration cycle on PCI bus		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		Undefined		

Table 94: PCI.PDR

4.5.5 PCI access to wrapper

PCI.WCBAR

This register sets the location of all the wrapper configuration registers (not the PCI standard configuration registers which are accessible using PCI configuration accesses, in the target memory space).

PCI.WCBAR				0x007C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[15:0]	16	No	Reserved	RO
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		
ADDR	[28:16]	13	No	Base address of wrapper configuration registers	RW
	Operation		Defines bits 16 to 28 of the lowest address of the wrapper configuration registers Can be set to any multiple of 64 kBytes up to the limit in the LSR		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
-	[31:29]	3	No	Reserved	RO
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 95: PCI.WCBAR

PCI.LOCCFG_UNLOCK

This register controls write access, from the PCI bus, to the wrapper configuration registers (not the PCI standard configuration registers). Write access to the local registers from SuperHyway is unaffected by the content of this register.

Since the register is itself located within the local register space, it is treated as a special case and will be write accessible from PCI even when bit 0 is cleared that is when all other local registers are read-only.

PCI.LOCCFG_UNLOCK				0x0034	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
ENABLE	0	1	No	Write access from PCI bus to wrapper configuration registers	RW
	Operation		0: Write access to wrapper registers from PCI bus disabled 1: Write access to wrapper registers from PCI bus enabled In both cases read access of wrapper registers from PCI bus is enabled.		
	Read		Returns current value		
	Write		When bits [31:24] are 0x5A: Updates current value only Other values for bits [31:24]: Write ignored		
	Hard reset		0 (disabled)		
-	[31:1]	31	No	-	RO
	Operation		-		
	Read		Returns 0		
	Write		Ignored (see notes above with regard to top 8 bits)		
	Hard reset		0		

Table 96: PCI.LOCCFG_UNLOCK

Note, it is only possible to write this register when the value of the top byte, that is bits [31:24] of the write data is 0x5A. Other values cause the write to be ignored.

4.5.6 Enhanced memory region mapping

PCI region 0 to 7 base address registers

These registers set the location of each memory mapped region in the target's PCI memory space. Only bits which correspond to 0's in the region's RSR register have any effect on the PCI address.

PCI_RBAR[n] where n = 0 to 7				0x01[n]0	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[15:0]	16	No	Reserved	RO
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		
ADDR	[28:16]	13	No	Base address of region in PCI space	RW
	Operation		PCI address falls within this region if upper bits match these in locations where the corresponding bit in PCI.RSR = 0		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
-	[31:29]	16	No		RO
	Operation				
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 97: PCI.RBAR[n]

PCI region 0 to 7 space registers

These registers set the size of each memory mapped region in the target's PCI memory space, and in local space. The size must be power of 2, and be ≥ 64 Kbytes and ≤ 512 Mbytes. The bits set to 1 in this register mask out the region's RBAR register, and allow bits in the translated local address to be given by the external address. Bit 0 acts as an enable for the region. Setting this bit to 0 causes accesses to this region to have no effect.

PCI.RSR[n] where n = 0 to 7				**0x01[n]4**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
ENABLE	0	1	No	Enable PCI accesses to region	RW
	Operation		0: Access to addresses in region disabled 1= Access to addresses in region enabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0 (disabled)		
-	[15:1]	15	No	Reserved	RO
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 98: PCI.RSR[n]

PCI.RSR[n] where n = 0 to 7				**0x01[n]4**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
SPACE	[28:16]	13	No	Set size of region (must be power of 2)	RW
	Operation		Specifies window occupied by PCI addresses and local addresses for region Only valid values (as shown in separate table) must be used. If bit is 0 corresponding bit in local address comes from PCI.LAR[N] If bit is 1 corresponding bit in local address comes from external address		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
-	[31:29]	3	No	Reserved	RO
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 98: PCI.RSR[n]

Only certain values, as shown in the table below, are legal for the RSR registers.

Region size	RSR value
Disabled	0x0000 0000
64 kBytes	0x0000 0001
128 kBytes	0x00010001
256 kBytes	0x00030001
512 kBytes	0x00070001
1 MBytes	0x000F0001
2 MBytes	0x001F0001
4 MBytes	0x003F0001
8 MBytes	0x007F0001
16 MBytes	0x00FF0001
32 MBytes	0x01FF0001
64 MBytes	0x03FF0001
128 MBytes	0x07FF0001
256 MBytes	0x0FFF0001
512 MBytes	0x1FFF0001

Table 99: Allowed values for RSR registers

PCI region 0 to 7 local address registers

These registers set the location of each memory mapped region in the local memory space. Each register replaces the uppermost bits of the external address so as to provide a new external address mapped to SuperHyway. Note that the memory regions have a simple fixed priority, so that if they overlap, only the highest priority RLAR actually translates the address for SuperHyway.

PCI_RLAR[n] where n = 0 to 7				**0x01[n]8**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[15:0]	16	No	Reserved	RO
	Operation		Reserved		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		
ADDR	[28:16]	13	No	Base address of region in local space	RW
	Operation		When the region is enabled and has highest priority, these bits become bits 16 to 28 of the local address in locations where the corresponding bit in the PCI.RSR = 0. When the region is disabled or has lower priority, these bits have no effect.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		

Table 100: PCI.RLAR[n]

PCI_RLAR[n] where n = 0 to 7				**0x01[n]8**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:29]	3	No	Base address of region in local space	RW
	Operation		When the region is enabled and has highest priority, bits of this register become bits 29 to 31 of the local address. When the region is disabled or has lower priority, these bits have no effect.		
	Read		Returns current value		
	Write		Returns current value		
	Hard reset		0		

Table 100: PCI.RLAR[n]

A table showing the relative priorities of overlapping translation regions is included below for reference. Note that all RLARs have a higher priority than local configuration register space.

RLAR	0	1	2	3	4	5	6	7	Local cfg
Priority	Highest	-------------------------------> Descending ------------------------------>							Lowest

Table 101: Translation priority for RLAR registers

4.5.7 CSR register bank

PCI vendor ID

This register identifies the manufacturer of the device. Valid vendor identifiers are allocated by PCI SIG to ensure uniqueness. ST's vendor ID is 4170 (decimal). The ID is hardware fixed and cannot be changed.

PCI.VID				0x10000	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_VID	[0:15]	16	-	PCI vendor ID	RO
	Operation		Identifies STMicroelectronics as the vendor of this device		
	Read		0x104A		
	Write		-		
	Hard reset		0x104A		

Table 102: PCI.VID

PCI device ID

This register identifies the device.

PCI.DID				0x10002	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_DID	[0:15]	16	-	PCI device ID	RO
	Operation		Hardware fixed		
	Read		0x0000		
	Write		-		
	Hard reset		0x0000[A]		

Table 103: PCI.DID

A. Check the datasheet for the value for any particular implementation.

PCI command

The PCI command register provides coarse control over a device’s ability to generate and respond to PCI cycles. When 0 is written to this register, the device is logically disconnected from the PCI bus for all accesses except configuration accesses.

PCI.CMD				**0x10004**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_IOS	0	1	-	PCI I/O space	RW
	Operation		Controls a device’s response to I/O space accesses		
	Read		0: Device response disabled 1: Device allowed to respond to I/O space accesses		
	Write		Updates current value		
	Hard reset		0		
PCI_MS	1	1	-	PCI memory space	RW
	Operation		Controls a device’s response to memory space accesses		
	Read		Returns current value		
	Write		0: Device response disabled 1: Device allowed to respond to memory space accesses		
	Hard reset		0		
PCI_BM	2	1	-	PCI bus master	RW
	Operation		Controls a device’s ability to act as a master on the PCI bus		
	Read		0: Device disabled from generating PCI accesses 1: Device enabled as bus master		
	Write		-		
	Hard reset		0		

Table 104: PCI.CMD

PCI.CMD				0x10004	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_SC	3	1	-	PCI special cycles	RO
	Operation		Controls device's action on special cycle operations		
	Read		0: Device ignores all special cycle operations 1: Device allowed to monitor special cycle operation (not support)		
	Write		-		
	Hard reset		0		
PCI_MWIE	4	1	-	PCI memory write and invalidate enable	RO
	Operation		Enables the memory write and invalidate command.		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_VGAPS	5	1	-	PCI VGA palette snoop	RO
	Operation		Controls how VGA compatible graphics device handles accesses to VGA palette register This interface does not support VGA snoop.		
	Read		0		
	Write		-		
	Hard reset		0		

Table 104: PCI.CMD

PCI.CMD				0x10004	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PER	6	1	-	PCI parity error response	RW
	Operation		Controls the device's response to parity errors When the bit is set, the device must take its normal action when a parity error is detected. When the bit is 0, the device sets its detected parity error status bit (bit 15 in the status register) when an error is detected, but does not assert pci_perrn and continues normal operation.		
	Read		Returns current value		
	Write		0: No response parity error 1: Response parity error		
	Hard reset		0		
PCI_SC	7	1	-	PCI stepping control	RO
	Operation		Controls whether or not a device does address/data stepping		
	Read		Returns current value		
	Write		0: Address/data stepping disabled 1: Address/data stepping enabled		
	Hard reset		0		
PCI_SERRE	8	1	-	PCI SERR enable	RW
	Operation		Enables the NOTPCI_SERR driver Address parity errors are reported only if this bit and bit 6 are 1.		
	Read		Returns current value		
	Write		0: NOT_PCI_SERR output disabled 1: NOT_PCI_SERR output enabled		
	Hard reset		0		

Table 104: PCI.CMD

PCI.CMD				0x10004	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_FBBE	9	1	-	PCI fast back-to-back enable	RO
	Operation		Controls whether master can do fast back-to-back transactions to different device 0: Fast back-to-back transactions are only allowed to the same agent 1: Master is allowed to generate fast back-to-back transactions to different agent (not supported)		
	Read		0		
	Write		-		
	Hard reset		0		
-	[10:15]	6	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		

Table 104: PCI.CMD

PCI status

This status register is used to record status information for PCI bus related events. The definition of each of the bits is given in *Table 105*. A device may not need to implement all the bits, depending on device functionality. For instance, a device that acts as a target, but will never signal target abort, would not implement bit 11 PCI.STATUS.PCI_STA. Reserved bits are read-only and return 0 when read.

Reads to this register behave normally. Writes are slightly different in that bits can be reset, but not set. A 1 bit is reset whenever the register is written, and the write data in the corresponding bit location is a 1. For instance, to clear bit 14 and not affect any other bits, write the value 16’b0100_0000_0000_0000 to the register.

PCI.STATUS				**0x10006**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[0:3]	5	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_CL	4	1	-	PCI capabilities list	RO
	Operation		Indicates whether or not this device implements the pointer for a new capabilities linked list at offset 0x34 (optional)		
	Read		1		
	Write		-		
	Hard reset		1		

Table 105: PCI.STATUS

PCI.STATUS				0x10006	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_66C	5	1	-	PCI 66MHz capable	RO
	Operation		Indicates whether this device is capable of running at 66MHz		
	Read		1: 66 MHz capable 0: 33 MHz capable		
	Write		Updates current value (SuperHyway only)		
	Hard reset		1		
-	6	5	-	Reserved	RO
	Operation		Reserved		
	Read		0		
	Write		-		
PCI_FBBC	7	1	-	PCI fast back-to-back capable	RO
	Operation		Indicates whether the target is capable of accepting fast back-to-back transactions when the transactions are not to the same agent (optional)		
	Read		1		
	Write		-		
	Hard reset		1		

Table 105: PCI.STATUS

PCI.STATUS				0x10006	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_MDPE	8	1	-	Master data parity error	RW
	Operation	This bit is used to bus masters only. It is set when three conditions are met: The bus agent asserted NOT_PCI_PERR itself (on a read) or observed NOT_PCI_PERR asserted (on a write). The agent setting the bit acted as the bus master for the operation in which the error occurred. PCI.CMD.PCI_PER is set.			
	Read	Returns current value			
	Write	Clear this bit			
	Hard reset	0			
PCI_DEVSEL	[10:9]	2	-	DEVSEL timing	RO
	Operation	pci_devseln timing status 2’b00: Fast (not support) 2’b01: Medium 2’b10: Slow (not support) 2’b11: Reserved			
	Read	2’b01			
	Write	-			
	Hard reset	2’b01			

Table 105: PCI.STATUS

PCI.STATUS				0x10006	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_STA	11	1	-	PCI signaled target abort	RW
	Operation		This bit must be set by a target device when it terminates a transaction with target abort.		
	Read		0: No target abort sent 1: Target ended the transaction with target abort		
	Write		Clear this bit		
	Hard reset		0		
PCI_RTA	12	1	-	PCI received target abort	RW
	Operation		This bit must be set by a master device when its transaction is terminated with target abort.		
	Read		0: No target abort received 1: Master detected end of transaction by target abort		
	Write		Clear this bit		
	Hard reset		0		
PCI_RMA	13	1	-	PCI received master abort	RW
	Operation		This bit must be set by a master device when its transaction is terminated with master abort (except for special cycle).		
	Read		0: No master abort received 1: Master detected end of the transaction by master abort (except special cycle)		
	Write		Clear this bit		
	Hard reset		0		

Table 105: PCI.STATUS

PCI.STATUS				0x10006	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_SSE	14	1	-	PCI signaled system error	RW
	Operation		This bit must be set when the device asserts NOT_PCI_SERR.		
	Read		0: Device is not asserted NOT_PCI_SERR 1: Device is asserted NOT_PCI_SERR		
	Write		Clear this bit		
	Hard reset		0		
PCI_DPE	15	1	-	PCI detected parity error	RW
	Operation		This bit must be set by the device when it detects a parity error, even if parity error handling is disabled (as controlled by PCI.CMD.PCI_PER, bit 6 in the command register).		
	Read		0: Device is not detecting parity error 1: Device is detecting parity error		
	Write		Clear this bit		
	Hard reset		0		

Table 105: PCI.STATUS

PCI revision ID

This register specifies a device-specific revision identifier.

PCI.RID				0x10008	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_RID	[0:7]	8	-	Revision ID	RO
	Operation		Hardware fixed		
	Read		0x01[A]		
	Write		-		
	Hard reset		0x01[A]		

Table 106: PCI.RID

A. subject to confirmation in the datasheet.

PCI class code

This field is the class code for the PCI interface.

PCI.CLASS				0x10009	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_CLASS	[0:23]	24	-	Class code	RO
	Operation		Identifies the generic function of the device See PCI specification for semantics of this register		
	Read		Returns current value		
	Write		-		
	Hard reset		0x040000		

Table 107: PCI.CLASS

PCI cache line size

PCI.CLS				0x1000C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_CLS	[0:7]	8	-	Cache line size	RO
	Operation		Hard fixed		
	Read		Returns current value		
	Write		-		
	Hard reset		0x00		

Table 108: PCI.CLS

PCI latency timer

This register specifies, in units of PCI bus clocks, the value of latency timer for this PCI bus master.

PCI.MLT				0x1000D	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_MLT	[2:7]	6	-	Latency timer	RW
	Operation		Binding in the biggest acquisition time of PCI bus		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		6’b000000		
-	[0:1]	2	-	Reserved	RO
	Operation				
	Read		0		
	Write		-		
	Hard reset		2’b00		

Table 109: PCI.MLT

PCI header type

This byte identifies the layout of the second part of the predefined header (beginning at byte 10h in configuration space) and also whether the device contains multiple functions.

PCI.HDR				0x1000e	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_MFE	7	1	-	Multiple function enable	RO
	Operation		Hard fixed (no support for multiple function)		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_HDR	[6:0]	7	-	PCI header type	RO
	Operation		Hard fixed (no support for multiple function)		
	Read		7’b000 0000		
	Write		-		
	Hard reset		7’b000 0000		

Table 110: PCI.HDR

PCI BIST

The BIST function is not supported.

PCI.BIST				0x1000F	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_BISTC	7	1	-	BIST capable	RO
	Operation		0: Not capable 1: Capable		
	Read		0		
	Write		-		
	Hard reset		0		
-	[6:0]	7	-	Reserved	RO
	Operation		Reserved		
	Read		2’b00		
	Write		-		
	Hard reset		2’b00		

Table 111: PCI.BIST

PCI memory base address register 0

This register controls access to the local address space from devices on the PCI bus. Refer to *Memory read / write on page 172*.

PCI.MBAR[0]				**0x10010**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MSI	0	1	-	Memory space indicator	RO
	Operation		Indicates that this register applies to PCI memory space		
	Read		0		
	Write		-		
	Hard reset		0		
TYPE	[1:2]	2	-	Address location type	RO
	Operation		Indicates that this device may be located anywhere in a 64-bit address space		
	Read		2’b10		
	Write		-		
	Hard reset		2’b10		
PREFETCHABLE	3	1	-	Local address prefetchable	RO
	Operation		Indicates that the local address space has no side effect on reads The device returns all bytes on reads regardless of byte enables.		
	Read		1		
	Write		-		
	Hard reset		1		

Table 112: PCI.MBAR[0]

PCI.MBAR[0]				**0x10010**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[4:15]	12	No	Reserved	RO
	Operation		-		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0x000000		
BASE_ADDR	[16:31]	16	-	Memory base address	RW*
	Operation		Specifies the base address of PCI memory window 0		
	Read		Returns current value		
	*Write		The local space register (PCI.LSR[0]) indicates which of these bits are read/write and which are read-only. If the corresponding bit in PCI.LSR is 0 then the bit is read/write. If the corresponding bit in the PCI.LSR is 1 then the bit is read-only.		
	Hard reset		0x0000		

Table 112: PCI.MBAR[0]

Using the PCI.MBAR[0] register the PCI module can indicate to PCI configuration software the size of the PCI memory address space required.

PCI I/O base address register

This register packages the I/O space base address register of the PCI configuration register that is prescribed with PCI local specification.

Refer to *Section : I/O read/write on page 175* on accessing PCI I/O space.

PCI.IBAR				**0x00018**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_IOB1	[31:8]	14	-	I/O base address (upper 24 bits)	RW
	Operation		Specifies top 24 bits of all I/O accesses destined for this PCI device		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0x000000		
-	7	1	-	Reserved	RO
	Operation		Constant		
	Read		0		
	Write		-		
	Hard reset		7'b0000000		
PCI_IOSI	0	1	-	I/O space indicator	RO
	Operation		Constant.		
	Read		1		
	Write		-		
	Hard reset		1'b1		

Table 113: PCI.IBAR

PCI subsystem vendor ID

Refer to section about miscellaneous registers of PCI Local Specification Revision 2.2.

The ST40 hardware does not set or interpret this field.

PCI.SVID				0x1002C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_SVID	[0:15]	16	-	Subsystem vendor ID	RW[A]
	Operation		-		
	Read		Returns current value		
	Write		Updates current value (SuperHyway only)		
	Hard reset		0x0000		

Table 114: PCI.SVID

A. RW if PCI.CR.PCI_CFINT = 0
RO if PCI.CR.PCI_CFINT = 1

PCI subsystem ID

Refer to section about miscellaneous registers of PCI Local Specification Revision 2.2.

The ST40 hardware does not set or interpret this field.

PCI.SID				0x1002E	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_SVID	[15:0]	16	-	Subsystem ID	RW[A]
	Operation		-		
	Read		Returns current value		
	Write		Updates current value (SuperHyway only)		
	Hard reset		0x0000		

Table 115: PCI.SID

A. RW if PCI.CR.PCI_CFINT = 0
RO if PCI.CR.PCI_CFINT = 1

PCI capabilities pointer

This register is the expansion function pointer register of the PCI configuration register that is prescribed in the PCI Power Management Specification.

PCI.CP				0x10034	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_CP	[7:0]	8	-	Capabilities pointer	RO
	Operation		Offset address of expansion function register		
	Read		0xDC		
	Write		-		
	Hard reset		0xDC		

Table 116: PCI.CP

PCI interrupt line

Sets priority for INT[N] pins, select by PCI interrupt pin register.

PCI.INTLINE				0x1003C	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_INTLINE	[7:0]	8	-	Interrupt line	RW
	Operation	The ST40 does not interpret or set this field.			
	Read	Returns current value			
	Write	Updates current value			
	Hard reset	8’h00			

Table 117: PCI.INTLINE

PCI interrupt pin

The interrupt pin register identifies which interrupt pins the device uses.

PCI.INTPIN				0x1003D	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_INTPIN	[7:0]	8	-	Interrupt pin	RW
	Operation	Select interrupt pin 8’h00: Does not use interrupt pin 8’h01: Uses the NOT_PCI_INTA pin All other values are undefined.			
	Read	Returns current value			
	Write	Updates current value			
	Hard reset	0x01			

Table 118: PCI.INTPIN

PCI minimum grant

This register is not programmable.

PCI.MINGNT				0x1003E	
Field	Bits	Size	Volatile	Synopsis	Type
PCI_MINGNT	[7:0]	8	-	Minimum grant	RO
	Operation		Hard fixed		
	Read		8'h00		
	Write		-		
	Hard reset		8'h00		

Table 119: PCI.MINGNT

PCI maximum latency

This register is not programmable.

PCI.MAXLAT				0x1003f	
Field	Bits	Size	Volatile	Synopsis	Type
PCI_MAXLAT	[7:0]	8	-	Maximum latency	RO
	Operation		Hard fixed		
	Read		8'h00		
	Write		-		
	Hard reset		8'h00		

Table 120: PCI.MAXLAT

PCI TRDY time-out value

PCI.TRDYTIME				0x10040	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_TRDYTIME	[7:0]	8	-	Configuration TRDY time-out value	RW
	Operation		Sets number of PCI clocks the core as master waits for TRDY		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0x80		

Table 121: PCI.TRDYTIME

PCI retry time-out value

PCI.RETRYTIME				0x10041	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_TRDYTIME	[7:0]	8	-	Configuration retry time-out value	RW
	Operation		Sets the number of retries the core as master performs		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0x80		

Table 122: PCI.RETRYTIME

PCI capability identifier

The capability identifier, when read by system software as 01h indicates that the data structure currently being pointed to is the PCI power management data structure. Each function of a PCI device may have only one item in its capability list with PCI_CID set to 01h.

PCI.CID				0x100DC	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_CID	[7:0]	8	-	Capability identifier	RO
	Operation		Hard fixed		
	Read		8’h01		
	Write		-		
	Hard reset		8’h01		

Table 123: PCI.CID

PCI next item pointer

The next item pointer register describes the location of the next item in the function’s capability list. The value given is an offset into the function’s PCI configuration space. If the function does not implement any other capabilities defined by the PCI SIG for inclusion in the capabilities list, or if power management is the last item in the list, then this register must be set to 00h.

PCI.NIP				0x100DD	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_NIP	[7:0]	8	-	Next item pointer	RO
	Operation		Hard fixed		
	Read		8’h00		
	Write		-		
	Hard reset		8’h00		

Table 124: PCI.NIP

PCI power management capability

The power management capabilities register is a 16-bit read-only register which provides information on the capabilities of the function related to power management. The information in this register is generally static and known at design time. This register is not cleared by hard reset. This register must set when PCI.CR.PCI_CFINT is 0.

PCI.PMC				0x100DE	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PMES	[15:11]	5	-	PME support	RO
	Operation		Indicates the power states in which the function may assert NOT_PCI_PME A value of 0b for any bit indicates that the function is not capable of asserting the NOT_PCI_PME signal while in that power state. Bit(11) xxxx1b = NOT_PCI_PME can be asserted from D0 Bit(12) xxx1xb = NOT_PCI_PME can be asserted from D1 Bit(13) xx1xxb = NOT_PCI_PME can be asserted from D2 Bit(14) x1xxxb = NOT_PCI_PME can be asserted from D3 hot Bit(15) 1xxxxb = NOT_PCI_PME can be asserted from D3 cold		
	Read		5’b00000		
	Write		-		
	Hard reset		5’b00000		

Table 125: PCI.PMC

PCI.PMC				0x100DE	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_D2S	10	1	-	D2 support	RO
	Operation		If this bit is 1, this function supports the D2 power management state. Functions that do not support D2 must always return a value of 0 for this bit.		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_D1S	9	1	-	D1 support	RO
	Operation		0: Function does not support D1 power management state 1: Function supports D1 power management state		
	Read		0		
	Write		-		
	Hard reset		0		
-	[8:6]	4	-	Reserved	RES
	Operation		-		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_DSI	5	1	-	Device specific initialization	RO
	Operation		Hard fixed		
	Read		0		
	Write		-		
	Hard reset		0		

Table 125: PCI.PMC

PCI.PMC				0x100DE	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	4	4	-	Reserved	RES
	Operation		-		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_PMEC	3	1	-	PME clock	RO
	Operation		0: No PCI clock required for function to generate pci_pme_n 1: Function relies on the presence of PCI clock for pci_pme[n] operation Functions that do not support pci_pme_n generation in any state must return 0 for this field.		
	Read		0		
	Write		-		
	Hard reset		0		
PCI_PMV	[2:0]	3	-	Power management version	RO
	Operation		A value of 3’b001 indicates that this function complies with Revision 1.0 of the PCI Power Management Interface Specification.		
	Read		3’b001		
	Write		Updates current value (SuperHyway only)		
	Hard reset		3’b001		

Table 125: PCI.PMC

PCI power management control/status

This 16-bit register is used to manage the PCI function’s power management state as well as to enable and monitor PMEs.

PCI.PMCSR				0x100E0	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PMES	15	1	-	PME status	RW*
	Operation		Set when the function would normally assert the NOT_PCI_PME signal independent of the state of the PCI.PMES.PCI_PMEEN bit. This bit defaults to 0 if the function does not support NOT_PCI_PME generation from D3 cold. If the function supports NOT_PCI_PME from D3 cold, then this bit is sticky and must be explicitly cleared by the operating system each time the operating system is initially loaded.		
	Read		Returns current value		
	Write		0: No effect 1: Clear the bit and cause the function to stop asserting a NOT_PCI_PME (if enabled)		
	Hard reset		1’b0		
PCI_DSC	[14:13]	2	-	Data scale	RO
	Operation		2’b00: Unknown 2’b01: 0.1 * PCI.PCDD (watts) 2’b10: 0.01 * PCI.PCDD (watts) 2’b11: 0.001 * PCI.PCDD (watts)		
	Read		Returns current value		
	Write		Updates current value (SuperHyway only)		
	Hard reset		2’b00		

Table 126: PCI.PMCSR

PCI.PMCSR				0x100E0	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_DSL	[12:9]	4	-	Data select	RO
	Operation		4’b0000: D0 power consumed		
	Read		0		
	Write		-		
	Hard reset		4’b0000		
PCI_PMEEN	8	1	-	PME enable	RW
	Operation		0: pci_pmen assertion disabled 1: Function enabled to assert pci_pmen. This bit defaults to 0 if the function does not support NOT_PCI_PME generation from D3 cold. If the function supports NOT_PCI_PME from D3 cold, then this bit is sticky and must be explicitly cleared by the operating system each time it is initially loaded. Functions that do not support NOT_PCI_PME generation from any D-state, may hardwire this bit to be read-only always returning 0 when read by system software.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1’b0		

Table 126: PCI.PMCSR

PCI.PMCSR				0x100E0	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PS	[1:0]	2	-	Power state	RW
	Operation		Determines the current power state of a function and sets the function into a new power state The definition of the field values is given below. 2’b00: D0 2’b01: D1 2’b10: D2 2’b11: D3 hot Only states D0 and D3 are supported. If software attempts to write an unsupported, optional state to this field, the write operation must complete normally on the bus. However, the data is discarded and no state change occurs.			
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		2’b00		
-	[7:2]	6	-	Reserved	RO
	Operation		Hard fixed		
	Read		6’b00 0000		
	Write		-		
	Hard reset		6’b00 0000		

Table 126: PCI.PMCSR

PCI PMCSR bridge support extension

This register supports PCI bridge specific functionality and is required for all PCI-to-PCI bridges. This register is not used by the ST40.

PCI.PMCSR_BSE				**0x100E2**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_BPCCEN	7	1	-	Bus power and clock control enable	RO
	Operation	When the bus power/clock control mechanism is disabled, the bridge’s PMCSR power state field cannot be used by the system software to control the power or clock of the bridge’s secondary bus.			
	Read	Returns current value			
	Write	-			
	Hard reset	1’bx			
PCI_B2B3N	6	1	-	B2/B3 support for D3 hot	RO
	Operation	The state of this bit determines the action that is to occur as a direct result of programming the function to D3 hot. 0: Secondary bus has its power removed (B3) 1: Secondary bus’s PCI clock is stopped (B2) This bit is only meaningful if bit 7 (PCI_BPCCEN) is set to 1.			
	Read	Returns current value			
	Write	-			
	Hard reset	1’bx			

Table 127: PCI.PMCSR_BSE

PCI.PMCSR_BSE				0x100E2	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[5:0]	6	-	Reserved	RO
	Operation		Hard fixed		
	Read		5’b0 0000		
	Write		-		
	Hard reset		5’b0 0000		

Table 127: PCI.PMCSR_BSE

PCI power consumption/dissipation data

The data register is an optional, 8-bit read-only register that provides a mechanism for the PCI module to report state dependent operating data such as power consumed or heat dissipation. Typically the data returned through the data register is a static copy (a look-up table, for example) of the function's worst case DC characteristics data sheet. This data, when made available to system software, could then be used to intelligently make decisions about power budgeting and cooling requirements. Any type of data could be reported through this register, but only power usage is defined by this version of the specification.

If the data register is present in a particular implementation then the PCI.PMCSR.PCI_DSL and PCI.PMCSR.PCI_DSC fields must also be implemented. If this register is not used then a value of 0 should always be returned by this register as well as for the PCI.PMCSR.PCI_DSL and PCI.PMCSR.PCI_DSC fields.

Software may check for the presence of the data register by writing different values into the PCI.PMCSR.PCI_DSL field, looking for non-zero return data in the data register or PCI.PMCSR.PCI_DSC field.

Any non-zero data register or PCI.PMCSR.PCI_DSL read data indicates that the data register complex has been implemented. Since PCI.PMCSR.PCI_DSL is a 4-bit field, an exhaustive presence detection scan requires 16 read/write operations to the PCI.PMCSR.PCI_DSL field, plus the data register or PCI.PMCSR.PCI_DSC field respectively.

Please refer to the datasheet for power dissipation figures.

PCI.PCDD				0x100E3	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PCI_PCDD	[7:0]	8	-	Power consumption/dissipation data	RW[A]
	Operation		Reports the state dependent data requested by the PCI.PMCSR.PCI_DSL field The value of this register is scaled by the value reported by the PCI.PMCSR.PCI_DSC field.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		8’h00		

Table 128: PCI.PCDD

A. Implementations may have this register read-only.

4.6 Pin list

Note: The pinout for a particular implementation may be different from that below see the datasheet for details.

Signal	Pins	Type	Description
PCI_AD[31:0]	32	Tristate I/O	PCI address/data bus Time-multiplexed address and data bus Each bus transaction consists of an address phase followed by one or more data phases.
NOT_PCI_C/BE[3:0]	4	Tristate I/O	PCI command/byte enable Time multiplexed bus command and byte enables Selects the type of transaction during the address phase and the byte enables during the data phases
PCI_PAR	1	Tristate I/O	PCI parity signal Generates and checks even parity across PCI_AD[31:0] and NOT_PCI_C/BE[3:0]
PCI_CLK	1	Input	PCI clock Provides timing for all transactions on the PCI bus
NOT_PCI_RST	1	I/O	PCI reset control Operates as input in satellite mode and output in host mode
NOT_PCI_FRAME	1	Tristate I/O	PCI Frame Driven by the current initiator and indicates the start and duration of a transaction
NOT_PCI_TRDY	1	Tristate I/O	PCI target ready Driven by the selected target. Is indicates the target's ability to complete the current data phase of the transaction

Signal	Pins	Type	Description
NOT_PCI_IRDY	1	Tristate I/O	PCI initiator ready Driven by the current bus master During a write, indicates that valid data is present on the AD[31:0] lines During a read indicates the master is ready to accept data
NOT_PCI_STOP	1	Tristate I/O	PCI stop Driven by the selected target to stop the current transaction
PCI_IDSEL	1	Input	PCI idsel Input to the PCI device to select for configuration cycles (only for satellite mode)
NOT_PCI_DEVSEL	1	Tristate I/O	PCI device select Indicates the driving device has decoded its address as the target As an input indicates that a device has been selected
NOT_PCI_INTA	1	Tristate I/O	Interrupt A Indicates a PCI device is requesting an interrupt in host bus bridge mode Output to request an interrupt in satellite mode
NOT_PCI_REQ[3:1]	4	Input	PCI bus request (only in host bus bridge mode)
NOT_PCI_GNT[3:1]	4	Output	PCI bus grant (only in host bus bridge mode)
NOT_PCI_REQ0	1	I/O	PCI bus request (input for host bus bridge mode, output for satellite mode)
NOT_PCI_GNT0	1	I/O	PCI bus grant (output for host bus bridge mode, input for satellite mode)
NOT_PCI_SERR	1	I/O	PCI system error

Signal	Pins	Type	Description
NOT_PCI_PERR	1	I/O	PCI parity error
PCI_CLKSEL	1	Input	PCI clock select 0: Internal PCI bridge move by internal clock 1: Internal PCI bridge move by PCI_CLK
PCI_HOST	1	Input	Strapping pin Sets PCI bridge mode 0: Satellite mode 1= Host bus bridge mode, sampled on the rising edge of reset_n

4.7 References

For more background on PCI, see:

- PCI Local Bus Specification Revision 2.2 (available through the PCI Special Interest Group (PCI SIG), http://www.pcisig.com),
- PCI Bus Power Management Interface Specification revision 1.1 (available through the PCI SIG, http://www.pcisig.com),
- PCI System Architecture (Tom Shanley, Don Anderson, MindShare, Inc., http://www.mindshare.com)

5 External microprocessor interface (EMPI)[1]

5.1 Overview

The external microprocessor interface (EMPI) module creates an efficient high performance communications channel for external devices in the system to access both internal and other external memory mapped devices mapped within the device.

It allows other devices to communicate using the MPX interface protocol which provides a pin efficient packet based interface operating at up to 400 Mbytes/s, which may be shared with other memory components in the system.

It provides mechanisms to reduce read latency costs with up to four independent read ahead buffers, and allows the entire ST40 memory space to be mapped on to an independent external address using a number of programmable windows. It also includes support for efficient use of multiple DMA channels, the ability to build communicating software processes or off-chip device drivers.

1. This chapter describes the EMPI interface implemented on production parts in the ST40 family. Pre production parts do not have the entire FMI feature set implemented and users of these parts should refer to revision B of this document.

5.2 Features

The main features of the EMPI are:

- 32 bit MPX satellite, target-only interface,
- synchronous operation at MPX clock speed, capable of 100 MHz,
- SuperHyway interface: one type 3 initiator plug and one type 1 target plug,
- four dedicated DMA channels each with 32-byte write FIFOs and two 32-byte read-ahead buffers:
  - DACK external selectors drive each DMA channel,
  - DREQ/DRACK handshakes available for each DMA channel,
  - automatic read buffer invalidate for each DMA channel on write address hit (for coherency)
  - read buffer invalidate for each DMA channel under software control,
  - read-ahead disable for each DMA channel.
- eight programmable regions to map MPX accesses into 4 Gbytes of SuperHyway space:
  - any region may use any DMA channel (if a DMA is not using it),
  - read or write buffers can be bypassed selectively on any region for I/O accesses.

The EMPI does not support the following:

- single address mode DMA,
- linear address mode DMA (all bursts wrap on burst-size boundary as per MPX spec),
- data coherency checking and invalidation across different DMA channels.

5.3 Register address map

The following table summarizes the registers used to program the EMPI unit. All the registers are located on 64-bit boundaries. Although in all cases no more than 32 bits are defined, these always correspond to the lower 4 bytes of the 64-bit location.

Register name	Description	Type	Address offset	Access size
EMPI.VCR_STATUS	Module status, see *Table 131 on page 294*	RW	0x0000	32
EMPI.VCR_VERSION	Module version, see *Table 132 on page 296*	RO	0x0008	32
EMPI.SYSTEM	Soft reset, see *Table 133 on page 297*	WO	0x0010	32
EMPI.ISTATUS	Interrupt status, see *Table 134 on page 298*	RW	0x0018	32
EMPI.IMASK	Interrupt mask, see *Table 135 on page 299*	RW	0x0020	32
EMPI.MPXCFG	External MPX interface configuration, see *Table 136 on page 300*	RW	0x0028	32
EMPI.DMAINV	Invalidate DMA read buffers, see *Table 137 on page 302*	WO	0x0030	32
Reserved			0x0038 to 0x0078	-
EMPI.DMACFG0	DMA buffer channel 0 configure, see *Table 138 on page 304*	RW	0x0080	4
EMPI.DMACFG1	DMA buffer channel 1 configure, see *Table 138 on page 304*	RW	0x0088	4

Table 129: EMPI registers

Register name	Description	Type	Address offset	Access size
EMPI.DMACFG2	DMA buffer channel 2 configure, see *Table 138 on page 304*	RW	0x0090	4
EMPI.DMACFG3	DMA buffer channel 3 configure, see *Table 138 on page 304*	RW	0x0098	4
Reserved			0x00A0 to 0x00F8	-
EMPI.DSTATUS0	DMA buffer channel 0 status, see *Table 139 on page 306*	RO	0x0100	32
EMPI.DSTATUS1	DMA buffer channel 1 status, see *Table 139 on page 306*	RO	0x0108	32
EMPI.DSTATUS2	DMA buffer channel 2 status, see *Table 139 on page 306*	RO	0x0110	32
EMPI.DSTATUS3	DMA buffer channel 3 status, see *Table 139 on page 306*	RO	0x0118	32
Reserved			0x0120 to 0x01F8	-
EMPI.RBAR0	MPX base address for region 0, see *Table 140 on page 307*	RW	0x0200	32
EMPI.RSR0	Region 0 size + buffer assign, see *Table 141 on page 308*	RW	0x0208	32
EMPI.RLAR0	Local base address for region 0, see *Table 142 on page 312*	RW	0x0210	32
Reserved			0x0218	-

Table 129: EMPI registers

Register name	Description	Type	Address offset	Access size
EMPI.RBAR1	MPX base address for region 1, see *Table 140 on page 307*	RW	0x0220	32
EMPI.RSR1	Region 1 size + buffer assign, see *Table 141 on page 308*	RW	0x0228	32
EMPI.RLAR1	Local base address for region 1, see *Table 142 on page 312*	RW	0x0230	32
Reserved			0x0238	-
EMPI.RBAR2	MPX base address for region 2, see *Table 140 on page 307*	RW	0x0240	32
EMPI.RSR2	Region 2 size + buffer assign, see *Table 141 on page 308*	RW	0x0248	32
EMPI.RLAR2	Local base address for region 2, see *Table 142 on page 312*	RW	0x0250	32
Reserved			0x0258	-
EMPI.RBAR3	MPX base address for region 3, see *Table 140 on page 307*	RW	0x0260	32
EMPI.RSR3	Region 3 size + buffer assign, see *Table 141 on page 308*	RW	0x0268	32
EMPI.RLAR3	Local base address for region 3, see *Table 142 on page 312*	RW	0x0270	32
Reserved			0x0278	-

Table 129: EMPI registers

Register name	Description	Type	Address offset	Access size
EMPI.RBAR4	MPX base address for region 4, see *Table 140 on page 307*	RW	0x0280	32
EMPI.RSR4	Region 4 size + buffer assign, see *Table 141 on page 308*	RW	0x0288	32
EMPI.RLAR4	Local base address for region 4, see *Table 142 on page 312*	RW	0x0290	32
Reserved			0x0298	-
EMPI.RBAR5	MPX base address for region 5, see *Table 140 on page 307*	RW	0x02A0	32
EMPI.RSR5	Region 5 size + buffer assign, see *Table 141 on page 308*	RW	0x02A8	32
EMPI.RLAR5	Local base address for region 5, see *Table 142 on page 312*	RW	0x02B0	32
Reserved			0x02B8	-
EMPI.RBAR6	MPX base address for region 6, see *Table 140 on page 307*	RW	0x02C0	32
EMPI.RSR6	Region 6 size + buffer assign, see *Table 141 on page 308*	RW	0x02C8	32
EMPI.RLAR6	Local base address for region 6, see *Table 142 on page 312*	RW	0x02D0	32
Reserved			0x02D8	-

Table 129: EMPI registers

Register name	Description	Type	Address offset	Access size
EMPI.RBAR7	MPX base address for region 7, see *Table 140 on page 307*	RW	0x02E0	32
EMPI.RSR7	Region 7 size + buffer assign, see *Table 141 on page 308*	RW	0x02E8	32
EMPI.RLAR7	Local base address for region 7, see *Table 142 on page 312*	RW	0x02F0	32
Reserved			0x02F8 to 0xFFFF	-

Table 129: EMPI registers

5.4 Operation

Figure 52 gives a structural overview of the EMPI design.

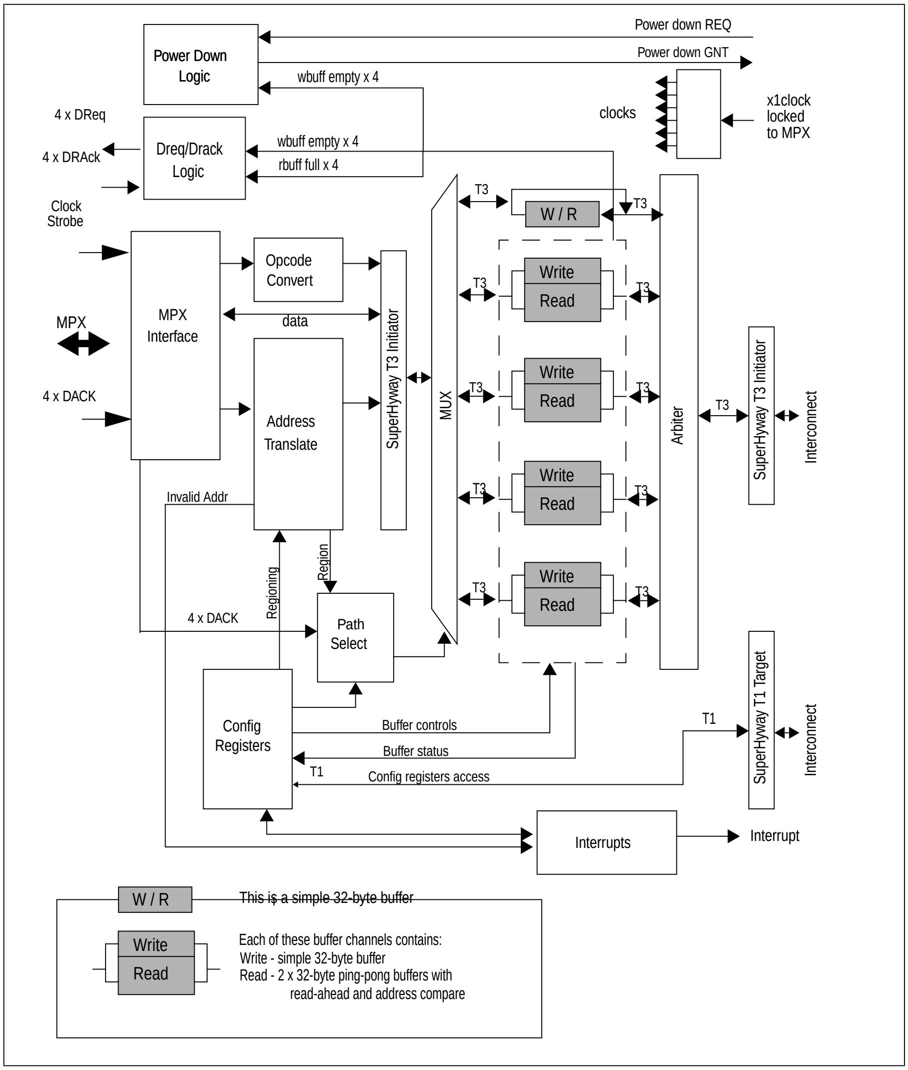


Figure 52: EMPI block overview

5.4.1 MPX interface

The MPX interface provides standard interface to MPX bus connection and protocol. The interface also contains a 32-bit buffer for off-loading data from the MPX bus. This interface deals with the MPX protocol and passes opcode, address, and data on.

5.4.2 Address translation

Eight programmable regions are provided to map addresses from MPX to the full 4 Gbytes space within SuperHyway. Each region can be disabled so that it has no effect using bit 0 of its RSR (see below).

Regions are programmed using BARs (base address register) with size masks in a similar way to PCI. 2 BARs per region specify the region's start, one in MPX space, and one in SuperHyway space. One region size register (RSR) allows each region to be programmed with a size from 64 Kbytes to 512 Mbytes in binary steps. Regions are restricted to start on boundaries which are integer multiples of their size (as with PCI).

Regions have a hard priority so that MPX addresses falling into more than one region are taken to be within the higher priority region only. Priority is highest for region 0 and lowest for region 7.

When an address is received which does not fall within any region, no translated address is produced, no transaction is passed to the SuperHyway interconnect, and an interrupt is raised. The access is accepted and MPX /RDY is still asserted in this case. If the MPX access is a read, invalid data is returned.

5.4.3 Data buffer channels

There are four data buffer channels, aimed at optimizing burst-reads and DMAs. Each channel consists of a 32-byte write FIFO, and two 32 byte read buffers, the latter arranged to ping-pong and offer read-ahead capability.

Write buffer operation

The write FIFO is 32 bits wide and 4 bits for byte enables. As each complete packet arrives from MPX a request to send it is passed to the SuperHyway arbiter.

The write FIFO does not consolidate writes into fixed size stores, it simply presents MPX write bursts in the size given by the MPX opcode. Consolidation could be a future performance enhancement.

Read buffer operation

The two read buffers are identical, and each has an associated register which stores the following information about the data it currently contains:

- start address of last burst request (all bursts into read buffers are 32-byte length),
- valid flag: true if buffer is not filling, and has not been invalidated.

Ping-pong buffers are used for reading ahead to allow one buffer to deliver data while the next fills.

Every read arriving at a buffer channel has address checked against the contents of either of the read buffers. If a match is found in buffer A, for example, and all the data requested by that access is within buffer A, and it is valid, the data is taken from the buffer. If read-ahead is enabled, then buffer B immediately requests data starting at buffer A start address plus 32 bytes. Once buffer B has refilled it is ready for address comparison with the next access. If no match is found with the next read, then neither buffer contains the desired data. In this case buffer A (the oldest) fills with 32 bytes of data from the requested address upward, while buffer B reads 32 bytes ahead. If read-ahead is disabled, then only one buffer is used.

Software may force the buffers to invalidate so that refills must take place before any read data can be returned to MPX. This allows the purging of known stale data.

5.4.4 Association between buffer channels and external DMA signals

Each buffer channel is hard associated (that is in a non-programmable assignment) with one of the four external DMA control signal sets. These signal sets consist of DACK, DREQ and DRACK.

- DACK steers the current MPX transaction through the associated buffer channel. This mechanism allows external DMA initiators (for example, DMAC) to use one channel, thus making efficient use of the read-ahead buffers. This pin would typically be connected to the DACK pin on another chip

  A configuration bit for each channel can disable this signal, thus effectively ignoring any changes on that DACK[N] pin. Note that DACK must not be disabled while a DMA is in progress.

- DREQ can be programmed (using one module configuration bit for a buffer channel) to have two meanings:
  - write FIFO is empty: in this case it provides an early message to a DMA initiator that another write can be started,
  - current read buffer is full and valid: in this case it provides an early message to a DMA initiator that another read can be started (no guarantee can be given that the buffer contains the required data however).

  This signal can only have one function at a time and should be programmed to match the current DMA data flow. The function can be disabled altogether if required using a module configuration bit.
- DRACK: handshake from the DMA initiator to signal the sampling of DREQ. This is used to return the current DREQ state to inactive, ready for an update on the next active condition (see above). If an initiator supports DRACK, then it can be used to provide more efficient DMA data flow control.

  This function can be disabled by a module configuration bit in which case DREQ always reflects the current condition of the buffer.

5.4.5 Association between buffer channels and address regions

Each address region can be associated with a specific buffer channel by means of 2 bits of its module configuration register. This assignment allows non-DMA transactions to make use of the read-ahead enhancement. The assignment is overridden by the DACK control signal see *Section 5.4.4: Association between buffer channels and external DMA signals on page 290* corresponding to that channel.

The use of a buffer channel’s read or write paths can also be disabled individually so that accesses pass directly to the SuperHyway interconnect, with no buffering. This mode would be used for I/O reads (a peripheral for example). Using this feature, a region could be programmed, say, to use one of the write buffers (for posting multi-byte writes) but access read data directly (slow but essential for a peripheral’s status registers).

Table 130 illustrates the buffer channel used by any given transfer.

Active address region								DMA control signals				Buffer channel used
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**DACK0**	**DACK1**	**DACK2**	**DACK3**	
At least one active								1	X	X	X	0
At least one active								0	1	X	X	1
At least one active								0	0	1	X	2
At least one active								0	0	0	1	3
1	X	X	X	X	X	X	X	0	0	0	0	Specified in configuration for region 0
0	1	X	X	X	X	X	X	0	0	0	0	Specified in configuration for region 1
0	0	1	X	X	X	X	X	0	0	0	0	Specified in configuration for region 2
0	0	0	1	X	X	X	X	0	0	0	0	Specified in configuration for region 3
0	0	0	0	1	X	X	X	0	0	0	0	Specified in configuration for region 4
0	0	0	0	0	1	X	X	0	0	0	0	Specified in configuration for region 5
0	0	0	0	0	0	1	X	0	0	0	0	Specified in configuration for region 6
0	0	0	0	0	0	0	1	0	0	0	0	Specified in configuration for region 7
0	0	0	0	0	0	0	0	X	X	X	X	Illegal - no data path

Table 130: Buffer channel choice against transaction conditions.

Note: In all cases the address translation is handled by the lowest numbered active region.

5.4.6 Opcode convert

This block translates MPX opcodes to SuperHyway opcodes. All accesses wrap as specified for MPX.

5.4.7 Arbiter

The arbiter which handles all buffer drains and fills has five inputs and one output. The inputs are SuperHyway targets since only one transaction at any time is produced by each FIFO. The extra input forms a bypass path for I/O type accesses. The output is an SuperHyway initiator allowing interleaving and reordering of accesses.

The arbitration scheme is round-robin, so that all accesses have a fair chance.

The arbiter should also, if possible, attempt to keep read and read-ahead burst requests together, as both normally fall within the same page of data memory. This improves LMI performance.

5.4.8 Configuration registers

All configuration registers within the EMPI block are read/write accessible from SuperHyway through a target interface.

5.4.9 Soft reset

A soft reset function is provided which returns all FSMs and address counters to their hard reset state.

5.4.10 Shut down

A shut-down request signal is provided which, when active, causes monitoring of the empty status of the EMPI buffers. Once these buffers are empty, a shut-down grant signal is raised, signalling that it is safe to power-down. Note that system software must ensure that there is no new traffic into or out of the MPX interface before requesting shut down, otherwise data may be lost.

5.5 Register definition

5.5.1 EMPI.VCR_STATUS

This register defines information available to the system, specifically debug, to determine how this module has interacted with the system, and if any erroneous requests have occurred during operation of that module.

EMPI.VCR_STATUS				0x0000	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
PERROR	[2:0]	3	Yes	Status of SuperHyway ports	RW
	Operation		Reports error status on EMPI SuperHyway ports An error has occurred when bit[n] = 1. Bit 0: Initiator port received an SuperHyway error response Bit 1: Target port returned an SuperHyway error response Bit 2: Target port accepted an SuperHyway access to an undefined location		
	Read		Returns current value		
	Write		0 written to bit [n]: Ignored 1 written to bit [n]: Reset bit [n] to 0		
	Hard reset		0		
-	[7:3]	5	No	Reserved	RW
	Operation		Reserved		
	Read		Undefined		
	Write		0: Ignored 1: Undefined		
	Hard reset		Undefined		

Table 131: EMPI.VCR_STATUS

EMPI.VCR_STATUS				0x0000	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MERROR	[8]	1	Yes	Module error	RW
	Operation		Reports error status of EMPI module When bit [n] =1 an error has occurred. Bit 8: A transaction using an illegally aligned address (see tables 10 and 11) was presented by MPX		
	Read		Returns current value		
	Write		0 written to bit [n]: Ignored 1 written to bit [n]: Reset bit [n] to 0		
	Hard reset		0		
-	[15:9]	7	No	Reserved	RW
	Operation		Reserved		
	Read		Undefined		
	Write		0: Ignored 1: Undefined		
	Hard reset		Undefined		
MOD_ID	[31:16]	16	No	Module identity	RO
	Operation		Value indicates the module type		
	Read		Returns 0x3D84 Each letter of EMP is encoded using a system where: Bits 4 to 0: First letter Bits 9 to 5: Second letter Bits 14 to 10: Third letter Letter A: 00000		
	Write		Ignored		
	Hard reset		0x3D84		

Table 131: EMPI.VCR_STATUS

5.5.2 EMPI.VCR_VERSION

This register is used to provide information about this block's version and requirements to the system.

EMPI.VCR_VERSION				**0x0008**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MOD_VER	[15:0]	16	No	Module version	RO
	Operation		Indicates the module version		
	Read		Returns 0x0001		
	Write		Ignored		
	Hard reset		0x0001		
MOD_SIZE	[31:16]	16	No	Module size	RO
	Operation		Module size defined as multiple of 64 Kbytes Read = 0x0001		
	Read		Returns current value		
	Write		Ignored		
	Hard reset		0x0001		

Table 132: EMPI.VCR_VERSION

5.5.3 EMPI.SYSTEM

This register is used to configure and control system level functions to the EMPI block.

EMPI.SYSTEM				0x0010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
SOFT_RST	0	1	Yes	Generate soft reset	WO
	Operation		When written resets all FSMs and counters to their hard reset state.		
	Read		Returns 0		
	Write		Updates current value 0: Ignored 1: Perform soft reset		
	Hard reset		0		
-	[31:1]	31	-	Reserved	Res
	Operation		Reserved		
	Read		Undefined		
	Write		0: No action 1: Undefined		
	Hard reset		Undefined		

Table 133: EMPI.SYSTEM

5.5.4 Interrupts

EMPI.ISTATUS

This register reflects the current state of all interrupt sources in the EMPI block. An interrupt only appears here if it is unmasked. When an unmasked interrupt occurs, the corresponding bit in this register is set to a 1. Writing back a 1 to any set bit results in the clearing of that bit.

EMPI.ISTATUS				**0x0018**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
UNMAP	0	1	Yes	Status of unmapped address interrupt	RW
	Operation		Set to 1 if an MPX address is received which does not map to any SuperHyway address.		
	Read		Returns current value 0: No interrupt since bit was last cleared 1: Interrupt caused by unmapped address error		
	Write		0: Ignored 1: Bit cleared to 0		
	Hard reset		0		
-	[31:1]	31	No	Reserved	RO
	Operation		Reserved		
	Read		Undefined		
	Write		0: No action 1: Undefined		
	Hard reset		Undefined		

Table 134: EMPI.ISTATUS

EMPI.IMASK

This register enables interrupt sources on a one-by-one basis. Bits are designed to correspond with those in the EMPI.ISTATUS register.

EMPI.IMASK				0x0020	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
ENABLE	0	1	No	Mask corresponding interrupt in istatus	RW
	Operation		Masks interrupt in istatus		
	Read		Returns current value		
	Write		Updates current value 0: Interrupt masked 1: Interrupt not masked		
	Hard reset		0		
-	[31:1]	31	No	Reserved	RO
	Operation		Reserved		
	Read		Undefined		
	Write		0: No action 1: Undefined		
	Hard reset		Undefined		

Table 135: EMPI.IMASK

5.5.5 MPX bus configuration

EMPI.MPXCFG

This register configures the external MPX interface (protocol, polarity of the ready signal and choice of the address map).

EMPI.MPXCFG				0x0028	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
TREADY	0	1	No	MPX TREADY polarity	RW
	Operation		Statically sets to MPX bus 0: T_READY is active low 1: T_READY is active high TREADY should only be updated once during post reset initialization and before any MPX traffic is directed to this EMPI. Updating at other times may cause undesired behavior of the MPX bus.		
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		

Table 136: EMPI.MPXCFG

EMPI.MPXCFG				0x0028	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
EN16	1	1	No	Enable MPX 16 byte burst	RW
	Operation		Statically enables decoding and processing of 16 byte bursts on MPX bus. This is an ST extension to the MPX specification and is **not supported by Hitachi MPX initiators.** 0: 16 byte burst reads and writes not decoded. 1: 16 byte burst reads and writes decoded. This should normally be updated once during post reset initialization and before any MPX traffic is directed to this EMPI. Further updates may only be made when the MPX bus is quiet. Updating at other times may cause deadlock of the MPX bus.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0 (16-byte bursts not decoded)		
-	[31:1]	31	No	Reserved	RO
	Operation		Reserved		
	Read		Undefined		
	Write		0: Ignored 1: Undefined		
	Hard reset		Undefined		

Table 136: EMPI.MPXCFG

5.5.6 DMA channel controls

EMPI.DMAINV

This register allows selective invalidation of the read buffers, so that a read-write-invalidate-read sequence can be used to produce fresh data, when the write and reads are to the same addresses.

Further bits allow selective purge of the write buffers. This provides the system software with a means of error recovery in the case where an incomplete write data packet was received from MPX. All purged data is lost.

EMPI.DMAINV				0x0030	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
INV0	0	1	Yes	Invalidate read data on channel 0	WO
	Operation		Invalidates all data held in the current buffer and the read ahead buffer in channel 0. 0: Ignored 1: Force invalidation		
	Read		Undefined		
	Write		1: Invalidates all pre-fetched read data in channel 0, for next read		
	Hard reset		0		
INV1	1	1	Yes	Invalidate read data on channel 1	WO
	Operation		Invalidates all data held in the current buffer and the read ahead buffer in channel 1 0: Ignored 1: Force invalidation		
	Read		Undefined		
	Write		1 invalidates all pre-fetched read data in channel 1, for next read.		
	Hard reset		0		

Table 137: EMPI.DMAINV

EMPI.DMAINV				0x0030	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
INV2	2	1	Yes	Invalidate read data on channel 2	WO
	Operation		Invalidates all data held in the current buffer and the read ahead buffer in channel 2 0: Ignored 1: Force invalidation		
	Read		Undefined		
	Write		1 invalidates all pre-fetched read data in channel 2, for next read.		
	Hard reset		0		
INV3	3	1	Yes	Invalidate read data on channel 3	WO
	Operation		Invalidates all data held in the current buffer and the read ahead buffer in channel 3 0: Ignored 1: Force invalidation		
	Read		Undefined		
	Write		1 invalidates all pre-fetched read data in channel 3, for next read.		
	Hard reset		0		
-	[31:4]	28	No	Reserved	RW
	Operation		Reserved		
	Read		Undefined		
	Write		0: No action 1: Undefined		
	Hard reset		Undefined		

Table 137: EMPI.DMAINV

EMPI.DMACFG0 to EMPI.DMACFG3

These registers configure DMA buffer channels 0 to 3 as required.

EMPI.DMACFG[n]				0x0080 + (n * 8)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
ENABLE	0	1	No	Enable external DMA controls	RW
	Operation		Enables the DACK and handshake signals for this channel 0: DACK and DRACK ignored for this channel 1: DACK forces accompanying MPX transaction to use this channel (DRACK also used if enabled) Update only when DMA initiator has been halted.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
DREQ	1	1	No	Select DREQ DMA request function	RW
	Operation		Selects between DREQ for writes and DREQ for reads 0: DREQ active on this channel’s write FIFO empty 1: DREQ active on this channel’s current read buffer full In all cases the DREQ active state is latched until cleared by DRACK when DRACK acknowledge handshake is enabled, otherwise DREQ is updated on every clock. Update only when channel DMA is disabled.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		

Table 138: EMPI.DMACFG[n]

EMPI.DMACFG[n]				0x0080 + (n * 8)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
DRACK	2	1	No	Enable DRACK handshake	RW
	Operation		Enables DRACK DMA return acknowledge handshake. 0: DRACK ignored 1: DRACK rising edge clears DREQ value. Update only when channel DMA is disabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
RA_DIS	3	1	No	Disable read ahead	RW
	Operation		0: Read buffer miss causes both read buffers to fill 1: Read buffer miss causes fetch of requested data only Update only when channel DMA is disabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
-	[31:4]	28	No	Reserved	RW
	Operation				
	Read		Returns undefined value		
	Write		0: ignored 1: undefined		
	Hard reset		Undefined		

Table 138: EMPI.DMACFG[n]

EMPI.DSTATUS0 to EMPI.DSTATUS3

These registers allow the CPU to read back the current status of the buffers in the DMA channels. This feature is provided primarily for MPX, EMPI and DMA debug.

EMPI.DSTATUS[n]				0x0080 + (n * 8)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
WBE	0	1	Yes	Write buffer empty status on channel 0	RO
	Operation		Reports status of write buffer in channel 0 0: Buffer not empty 1: Buffer empty		
	Read		Returns status		
	Write		Ignored		
	Hard reset		1		
RBF	1	1	Yes	Read buffer status on channel [n]	RO
	Operation		Reports status of read buffer in channel [n] 0: Buffer not full 1: Buffer full		
	Read		Returns status		
	Write		Ignored		
	Hard reset		0		
-	[31:2]	30	No	Reserved	RW
	Operation		Reserved		
	Read		Undefined		
	Write		0: Ignored 1: Undefined		
	Hard reset		Undefined		

Table 139: EMPI.DSTATUS[n]

5.5.7 Address translation and buffer channel assignment

EMPI.RBAR0 to EMPI.RBAR7

These registers set the location of each memory mapped region in the Target's MPX memory space. Only bits which correspond to 0s in the region's RSR register have any effect on the MPX address.

EMPI.RBAR[n]				0x02[n]0 (where n = 2, 4, 6, 8, A, C, E)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[15:0]	16	No	Reserved	RO
	Operation				
	Read		Returns 0		
	Write		Ignored		
	Hard reset		0		
ADDR	[28:16]	13	No	Base address of region in MPX space	RW
	Operation		MPX address falls within this region if upper bits match these in locations where the corresponding bit in EMPI.RSR = 0. Update this register only when region is disabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
-	[31:29]	16	No	Reserved	RO
	Operation		Reserved		
	Read		Undefined		
	Write		0: No action 1: Undefined		
	Hard reset		Undefined		

Table 140: EMPI.RBAR[n]

EMPI.RSR0 to EMPI.RSR7

These registers set the size of each memory mapped region in the target's MPX memory space, and in local space. The size must be power of 2, and be ≥ 64 Kbytes and ≤ 512 Mbytes. The bits set to 1 in this register mask out the region's RBAR register, and allow bits in the translated local address to be given by the external address. Bit 0 acts as an enable for the region. Setting this bit to 0 causes accesses to this region to have no effect.

EMPI.RSR[n]				**0x020[n]8 (where n = 0, 2, 4, 6, 8, A, C, E)**	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
ENABLE	0	1	No	Enable MPX accesses to region	RW
	Operation		0: Access to addresses in region disabled 1= Access to addresses in region enabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0 (disabled)		
BUFFER	[2:1]	2	No	Select buffer channel for this region	RW
	Operation		Assigns a buffer channel to an access passing to this region 00: Accesses go through buffer channel 0 01: Accesses go through buffer channel 1 10: Accesses go through buffer channel 2 11: Accesses go through buffer channel 3 This assignment gets overridden by an active DACK signal for the channel. Assignment also gets overridden by an active bypass bit. Update this register only when region is disabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		

Table 141: EMPI.RSR[n]

EMPI.RSR[n]				0x020[n]8 (where n = 0, 2, 4, 6, 8, A, C, E)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
WPASS	3	1	No	Bypass write buffers for this region.	RW
	Operation		Bypasses any channel's write buffer for an access passing to this region 0: Write accesses go through channel assigned by buffer bits 1: Write accesses go directly to SuperHyway The value in this bit has priority over the buffer bits Update this register only when region is disabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1		
RPASS	4	1	No	Bypass read buffers for this region.	RW
	Operation		Bypasses any channel's read buffer for an access passing to this region 0: Read accesses go through channel assigned by buffer bits 1: Read accesses go directly to SuperHyway The value in this bit has priority over the buffer bits Update this register only when region is disabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		1		

Table 141: EMPI.RSR[n]

EMPI.RSR[n]				0x020[n]8 (where n = 0, 2, 4, 6, 8, A, C, E)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[15:5]	11	No	Reserved	RO
	Operation		Reserved		
	Read		Undefined		
	Write		0: No action 1: Undefined		
	Hard reset		Undefined		
SPACE	[28:16]	13	No	Set size of region (must be power of 2)	RW
	Operation		Specifies window occupied by MPX addresses and local addresses for region If bit = 0, corresponding bit in local address comes from EMPI.LAR[N] If bit = 1, corresponding bit in local address comes from external address Update this register only when region is disabled Only valid values must be used: 0x00000000: Disabled 0x00000001: 64 Kbytes 0x00010001: 128 Kbytes 0x00030001: 256 Kbytes 0x00070001: 512 Kbytes 0x000F0001: 1 Mbytes 0x001F0001: 2 Mbytes 0x003F0001: 4 Mbytes 0x007F0001: 8 Mbytes 0x00FF0001: 16 Mbytes 0x01FF0001: 32 Mbytes 0x03FF0001: 64 Mbytes 0x07FF0001: 128 Mbytes 0x0FFF0001: 256 Mbytes 0x1FFF0001: 512 Mbytes		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		

Table 141: EMPI.RSR[n]

EMPI.RSR[n]				0x020[n]8 (where n = 0, 2, 4, 6, 8, A, C, E)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[31:29]	3	No	Reserved	RO
	Operation		Reserved		
	Read		Undefined		
	Write		0: No action 1: Undefined		
	Hard reset		Undefined		

Table 141: EMPI.RSR[n]

EMPI.RLAR0 to EMPI.RLAR7

These registers set the location of each memory mapped region in the local memory space. Each register replaces the uppermost bits of the external address so as to provide a new external address mapped to SuperHyway. Note that the memory regions have a simple fixed priority, so that if they overlap, only the highest priority RLAR actually translates the address for SuperHyway.

EMPI.RLAR[n]				0x02[n]0 (where n = 1, 3, 5, 7, 9, B, D, F)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
-	[15:0]	16	No	Reserved	RO
	Operation		Reserved		
	Read		Undefined		
	Write		0: No action 1: Undefined		
	Hard reset		Undefined		

Table 142: EMPI.RLAR[n]

EMPI.RLAR[n]				0x02[n]0 (where n = 1, 3, 5, 7, 9, B, D, F)	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
ADDR	[28:16]	13	No	Base address of region in local space	RW
	Operation		When the region is enabled and has highest priority, these bits become bits 16 to 28 of the local address in locations where the corresponding bit in the EMPI.RSR = 0. When the region is disabled or has lower priority, these bits have no effect. Update this register only when region is disabled		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0		
	[31:29]	3	No	Base address of region in local space	RW
	Operation		When the region is enabled and has highest priority, bits of this register become bits 29 to 31 of the local address. When the region is disabled or has lower priority, these bits have no effect.		
	Read		Returns current value		
	Write		Returns current value		
	Hard reset		0		

Table 142: EMPI.RLAR[n]

A table showing the relative priorities of overlapping translation regions is included below for reference.

RLAR	0	1	2	3	4	5	6	7
Priority	Highest	-----------------------------> Descending ---------------------------->						Lowest

Table 143: Translation priority for RLAR[n] registers

6 MPX arbiter (MPXARB)[1]

6.1 Overview

This register interface allows CPU access to the ST40 EMI subsystem MPX arbiter.

The MPX arbiter in the ST40 arbitrates and assigns bus mastership between the ST40 and one other external MPX bus master. This internal ST40 MPX arbiter can be disabled (by external mode pin or software) to allow the use of an external arbiter to support more than two MPX bus masters.

6.1.1 Arbitration priority

The MPX arbiter has two programmable modes of operation for arbitration when simultaneous requests are pending:

- priority arbitration with ST40 highest priority (default reset case),
- priority arbitration with ST40 lowest priority.

1. This chapter describes the MPX arbiter implemented on production parts in the ST40 family. Pre-production parts do not have the entire EMI feature set implemented and users of these parts should refer to revision B of this document.

6.1.2 Bus grant parking

The MPX arbiter supports four methods of bus ownership parking when no requests are pending. Bus grant parking reduces the time taken for a agent to obtain bus mastership if no other requests are pending.

- Park the MPX bus in idle state with no bus grant asserted.
- Park the MPX bus with bus grant asserted to the last agent to use the bus. This reduces arbitration time if the last user requests the bus again.
- Park the MPX bus with bus grant asserted to the ST40 EMI.
- Park the MPX bus with bus grant asserted to the external master.

6.1.3 External request blocking

A register bit is used to block external requests to the MPX arbiter. This bit is set at reset to allow the ST40 to have ownership of the MPX bus until it has booted and clears the external request block bit. This mechanism prevents bus contention when two masters try to boot from the same MPX or EMI bus by allowing the ST40 to be the boot master.

6.1.4 Options for external arbitration

The ST40 MPX bus arbiter can be disabled and the bus request and grant signals passed to an external arbiter. This external arbiter can be designed to support more than two MPX bus masters or implement a more flexible bus arbitration method. A programmable (1 to 16) latency register is provided to limit the number of sequential MPX transactions an agent can own on the bus. A new transaction is identified by assertion of the MPX_FRAME signal indicating a new address phase.

Note: The bus transaction counter is reset when the bus is idle (parked) or the bus ownership changes.

6.2 Address map

Table 144 shows the address map of the MPX arbiter registers.

Register name	Description	Type	Address	Access size
MPXARB.VCR	MPX arbiter version control register, see *Table 145 on page 318*	RO	EMPI + x8000	32
Reserved			8008 to 800F	-
MPXARB.CONTROL	Control operating mode of MPX arbiter, see *Table 146 on page 319*	RW	8010	32
Reserved			8018 to BFFF	-
MPXARB.DLLCONTROL	External DLL control register, see *Table 147 on page 323*	RW	C000	32
Reserved			C008 to C00F	-
MPXARB.DLLSTATUS	External DLL Status register, see *Table 148 on page 323*	RO	C010	32
Reserved			C018 to FFFF	-

Table 144: MPX arbiter register organization

6.3 Register definition

6.3.1 MPXARB.VCR

Version control register of MPX arbiter unit.

MPXARB.VCR				0x000	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MVERS	[16:31]	16	No	Module version	RO
	Operation		Indicates module version number		
	Read		Returns 0x0000		
	Write		Ignored		
	Hard reset		0		

Table 145: MPXARB.VCR

6.3.2 MPXARB.CONTROL

MPX arbiter control register.

MPXARB.CONTROL				0x1B138010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
HOST	0	1	No	MPX arbiter host/satellite configuration	RO
	Operation		Reflects the bus mastership for ST40 For example, this bit detects for software whether or not the arbiter has been enabled by mode pin.		
	Read		0: MPX arbiter is satellite (disabled via mode pin) 1: MPX arbiter is host (enabled via pin)		
	Write		Ignored		
	hard reset		MPX arbiter host/satellite mode pin		
EXTSLAVE	1	1	No	External master or slave	RW
	Operation		Reflects the bus mastership for external agent (for example, EMI in companion chip)		
	Read		0: External agent master 1: External agent slave		
	Write		Updates current value by-passing external mode pin		
	Hard reset		External agent master/slave mode pin		

Table 146: MPXARB.CONTROL

MPXARB.CONTROL				0x1B138010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
ENABLE	2	1	No	MPX arbiter enable bit	RW
	Operation		Software enables and disables MPX arbiter (only when ST40 host mode pin is high) When host mode pin is low the arbiter is automatically disabled.		
	Read		Returns current value		
	Write		Updates current value 0: Disabled 1: Enabled		
	Hard reset		0x1: Enabled		
BLOCK	3	1	No	External request blocking bit	RW
	Operation		When set any external request is ignored		
	Read		Returns current value		
	Write		Updates current value 0: External request considered by MPX arbiter 1: External request are blocked by the MPX arbiter		
	Hard reset		0x1= External requests blocked		

Table 146: MPXARB.CONTROL

MPXARB.CONTROL				0x1B138010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
MODE	[4:5]	2	No	MPX arbiter operating mode (arbitration policy)	RW
	Operation		Selects the arbitration policy of MPX arbiter		
	Read		Returns current value		
	Write		Updates current value 00: ST40 highest priority 01: ST40 lowest priority 10: Reserved 11: Reserved		
	Hard reset		0x00		
PARK	[6:7]	2	No	MPX bus parking method	RW
	Operation		Sets the MPX bus parking method when no requests are pending		
	Read		Returns current value		
	Write		Updates current value 00: Park bus in idle state (neither ST40 nor external agents drive the bus) 01: Park MPX bus with last user (If no access pending bus ownership is left to the last user) 10: Park bus to ST40 11: Park bus to external agent (only when external agent is master)		
	Hard reset		0x10		

Table 146: MPXARB.CONTROL

MPXARB.CONTROL				0x1B138010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
BUS TENURE	[8:10]	3	-	Set the maximum bus tenure for the bus owner	-
	Operation		When the current owner is using the bus and a request from another agent arrives, sets a time-out to release the bus and avoid a deadlock		
	Read		Returns current value		
	Write		000: 16 cycles 001= 32 cycles 010: 64 cycles 011: 128 cycles 100= 256 cycles 101= 512 cycles 111= Bus tenure time infinite (timer disabled) Other = Reserved		
	Hard reset		0x000		
-	[11:31]	21	-	Reserved	-
	Operation		Reserved		
	Read		Undefined		
	Write		Ignored		
	Hard reset		Undefined		

Table 146: MPXARB.CONTROL

6.3.3 MPXARB.DLLCONTROL

MPX arbiter DLL control register.

MPXARB.DLLCONTROL				0X1B13C000	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
DLLCONTROL	[0:31]	32	No	External DLL control	RW
	Operation		These bits are available for general configuration of external DLL. The content of this register is output from EMI SS. A detailed map of each bit meaning is dependent on external DLL.		
	Read		Returns current value		
	Write		Updates current value		
	Hard reset		0x0000		

Table 147: MPXARB.DLLCONTROL

6.3.4 MPXARB.DLLSTATUS

MPX arbiter DLL status register.

MPXARB.DLLSTATUS				0x1B13C010	
Field	**Bits**	**Size**	**Volatile**	**Synopsis**	**Type**
DLLSTATUS	[0:31]	32	No	External DLL status	RO
	Operation		Reports general status of external DLL A detailed map of each bit meaning is dependent on external DLL.		
	Read		Returns current value		
	Write		Ignored		
	Hard reset		Depends on DLL hard reset values		

Table 148: MPXARB.DLLSTATUS

Appendices

A Register address list

Block	Register name	Description	Address offset
EMI	CONFIGURATION	Information which defines the operation of each bank, see *Table 38 on page 123*	0x7F0000 to 0x7FFFFF
EMI	BANK0	Bank0 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	0x100 to 0x138
EMI	BANK1	Bank1 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	0x140 to 0x178
EMI	BANK2	Bank2 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	0x180 to 0x1B8
EMI	BANK3	Bank3 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	0x1C0 to 0x1F8

Table 149: Register address list

Block	Register name	Description	Address offset
EMI	BANK4	Bank4 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	0x200 to 0x238
EMI	BANK5	Bank5 configuration register set (EMICONFIGDATA[0:3] plus reserved space), see *Section 2.9.13: Configuration register formats on page 135*	0x240 to 0x278
EMI	CLKENABLE	Enable clock generation for different devices, see *Table 49 on page 134*	0x68
EMI	FLASHCLKSEL	Select clock speed for flash devices, see *Table 46 on page 131*	0x50
EMI	GENCFG	General purpose configuration register set, see *Table 41 on page 126*	0x28
EMI	LOCK	Lock register, see *Table 40 on page 125*	0x20
EMI	MPXCLKSEL	Select clock speed for MPX devices, see *Table 48 on page 133*	0x60
EMI	REFRESHINT	Refresh interval for SDRAM, see *Table 45 on page 130*	0x48
EMI	SDRAMCLKSEL	Select clock speed for SDRAM devices, see *Table 47 on page 132*	0x58
EMI	SDRAMINIT	Initialize for SDRAM, see *Table 44 on page 129*	0x40
EMI	SDRAMNOPGEN	Generate NOP commands during the initialization phase of SDRAM until SDRAMINIT is set, see *Table 42 on page 127*	0x30
EMI	SDRAMODEREG	SDRAM configuration data, see *Table 43 on page 128*	0x38

Table 149: Register address list

Block	Register name	Description	Address offset
EMI	STATUSCFG	Status register (configuration flags update), see *Table 38 on page 123*	0x10
EMI	STATUSLOCK	Lock register (configuration flags lock), see *Table 39 on page 124*	0x18
EMI buffer	BANK_0_TOP_ADDRESS	Most significant 8 bits of external memory bank 0, see *BANK_0_BASE_ADDRESS on page 154*	0x000
EMI buffer	BANK_1_TOP_ADDRESS	Most significant 8 bits of external memory bank 1, see *BANK_1_BASE_ADDRESS on page 155*	0x010
EMI buffer	BANK_2_TOP_ADDRESS	Most significant 8 bits of external memory bank 2, see *BANK_2_BASE_ADDRESS on page 155*	0x020
EMI buffer	BANK_3_TOP_ADDRESS	Most significant 8 bits of external memory bank 3, see *BANK_3_BASE_ADDRESS on page 156*	0x030
EMI buffer	BANK_4_TOP_ADDRESS	Most significant 8 bits of external memory bank 4, see *BANK_4_BASE_ADDRESS on page 156*	0x040
EMI buffer	BANK_5_TOP_ADDRESS	Most significant 8 bits of external memory bank 5, see *BANK_5_BASE_ADDRESS on page 157*	0x050
EMI buffer	BANK_ENABLED	Total number of enabled banks, see *BANK_ENABLED on page 157*	0x060
EMPI	DMACFG0	DMA buffer channel 0 configure, see *Table 138 on page 304*	0x0080
EMPI	DMACFG1	DMA buffer channel 1 configure, see *Table 138 on page 304*	0x0088

Table 149: Register address list

Block	Register name	Description	Address offset
EMPI	DMACFG2	DMA buffer channel 2 configure, see *Table 138 on page 304*	0x0090
EMPI	DMACFG3	DMA buffer channel 3 configure, see *Table 138 on page 304*	0x0098
EMPI	DMAINV	Invalidate DMA read buffers, see *Table 137 on page 302*	0x0030
EMPI	DSTATUS0	DMA buffer channel 0 status, see *Table 139 on page 306*	0x0100
EMPI	DSTATUS1	DMA buffer channel 1 status, see *Table 139 on page 306*	0x0108
EMPI	DSTATUS2	DMA buffer channel 2 status, see *Table 139 on page 306*	0x0110
EMPI	DSTATUS3	DMA buffer channel 3 status, see *Table 139 on page 306*	0x0118
EMPI	IMASK	Interrupt mask, see *Table 135 on page 299*	0x0020
EMPI	ISTATUS	Interrupt status, see *Table 134 on page 298*	0x0018
EMPI	MPXCFG	External MPX interface configuration, see *Table 136 on page 300*	0x0028
EMPI	RBAR0	MPX base address for region 0, see *Table 140 on page 307*	0x0200
EMPI	RBAR1	MPX base address for region 1, see *Table 140 on page 307*	0x0220
EMPI	RBAR2	MPX base address for region 2, see *Table 140 on page 307*	0x0240
EMPI	RBAR3	MPX base address for region 3, see *Table 140 on page 307*	0x0260
EMPI	RBAR4	MPX base address for region 4, see *Table 140 on page 307*	0x0280

Table 149: Register address list

Block	Register name	Description	Address offset
EMPI	RBAR5	MPX base address for region 5, see *Table 140 on page 307*	0x02A0
EMPI	RBAR6	MPX base address for region 6, see *Table 140 on page 307*	0x02C0
EMPI	RBAR7	MPX base address for region 7, see *Table 140 on page 307*	0x02E0
EMPI	RLAR0	Local base address for region 0, see *Table 142 on page 312*	0x0210
EMPI	RLAR1	Local base address for region 1, see *Table 142 on page 312*	0x0230
EMPI	RLAR2	Local base address for region 2, see *Table 142 on page 312*	0x0250
EMPI	RLAR3	Local base address for region 3, see *Table 142 on page 312*	0x0270
EMPI	RLAR4	Local base address for region 4, see *Table 142 on page 312*	0x0290
EMPI	RLAR5	Local base address for region 5, see *Table 142 on page 312*	0x02B0
EMPI	RLAR6	Local base address for region 6, see *Table 142 on page 312*	0x02D0
EMPI	RLAR7	Local base address for region 7, see *Table 142 on page 312*	0x02F0
EMPI	RSR0	Region 0 size + buffer assign, see *Table 141 on page 308*	0x0208
EMPI	RSR1	Region 1 size + buffer assign, see *Table 141 on page 308*	0x0228
EMPI	RSR2	Region 2 size + buffer assign, see *Table 141 on page 308*	0x0248
EMPI	RSR3	Region 3 size + buffer assign, see *Table 141 on page 308*	0x0268

Table 149: Register address list

Block	Register name	Description	Address offset
EMPI	RSR4	Region 4 size + buffer assign, see *Table 141 on page 308*	0x0288
EMPI	RSR5	Region 5 size + buffer assign, see *Table 141 on page 308*	0x02A8
EMPI	RSR6	Region 6 size + buffer assign, see *Table 141 on page 308*	0x02C8
EMPI	RSR7	Region 7 size + buffer assign, see *Table 141 on page 308*	0x02E8
EMPI	SYSTEM	Soft reset, see *Table 133 on page 297*	0x0010
EMPI	VCR_STATUS	Module status, see *Table 131 on page 294*	0x0000
EMPI	VCR_VERSION	Module version, see *Table 132 on page 296*	0x0008
LMI	MIM	Memory interface mode, see *Table 14 on page 36*	0x00 0008
LMI	PBS	Pin buffer strength, see	0x00 0020
LMI	SCR	SDRAM control, see *Table 15 on page 40*	0x00 0010
LMI	SDMR0	SDRAM mode register, see *Section 1.4.6: SDRAM row mode registers (LMI.SDMR[0:1]) on page 49*	0x8x xxxx
LMI	SDMR1	SDRAM mode register, see *Section 1.4.6: SDRAM row mode registers (LMI.SDMR[0:1]) on page 49*	0x9x xxxx
LMI	SDRA0	SDRAM row attribute, see *Table 17 on page 47*	0x00 0030
LMI	SDRA1	SDRAM row attribute, see *Table 17 on page 47*	0x00 0038

Table 149: Register address list

Block	Register name	Description	Address offset
LMI	STR	SDRAM timing, see *Table 16 on page 43*	0x00 0018
LMI	VCR	Version control register, see *Table 11 on page 30*	0x00 0000
MPXARB	CONTROL	Control operating mode of MPX arbiter, see *Table 146 on page 319*	8010
MPXARB	DLLCONTROL	External DLL control register, see *Table 147 on page 323*	C000
MPXARB	DLLSTATUS	External DLL Status register, see *Table 148 on page 323*	C010
MPXARB	VCR	MPX arbiter version control register, see *Table 145 on page 318*	EMPI + x8000
PCI	AINT	PCI arbiter interrupt register, see *Table 84 on page 218*	0x00040
PCI	AINTM	PCI arbiter interrupt mask register, see *Table 85 on page 221*	0x00044
PCI	AIR	PCI error address information register, see *Table 82 on page 215*	0x0002C
PCI	BIST	PCI BIST, see *Table 111 on page 258*	0x1000F
PCI	BMIR	PCI error information, register of bus master, see *Table 86 on page 224*	0x00048
PCI	CID	PCI capability ID, see *Table 123 on page 267*	0x100DC
PCI	CIR	PCI error command information register, see *Table 83 on page 216*	0x00030
PCI	CLASS	PCI class code, see *Table 107 on page 255*	0x10009

Table 149: Register address list

Block	Register name	Description	Address offset
PCI	CLS	PCI cache line size, see *Table 108 on page 256*	0x1000C
PCI	CMD	PCI command, see *Table 104 on page 246*	0x10004
PCI	CP	PCI capabilities pointer, see *Table 116 on page 263*	0x10034
PCI	CR	PCI control register, see *Table 77 on page 195*	0x00010
PCI	DID	PCI device ID, see *Table 103 on page 245*	0x10002
PCI	HDR	PCI header type, see *Table 110 on page 257*	0x1000E
PCI	IBAR	PCI I/O base address, see *Table 113 on page 261*	0x10018
PCI	INT	PCI interrupt register, see *Table 80 on page 205*	0x00024
PCI	INTLINE	PCI interrupt line, see *Table 117 on page 264*	0x1003C
PCI	INTM	PCI interrupt mask register, see *Table 81 on page 210*	0x00028
PCI	INTPIN	PCI interrupt pin, see *Table 118 on page 264*	0x1003D
PCI	IOBMR	PCI I/O space bank mask register, see *Table 93 on page 235*	0x00074
PCI	IOBR	PCI I/O space bank register, see *Table 89 on page 230*	0x00054
PCI	LAR[0]	PCI local address register 0, see *Table 79 on page 204*	0x0001C
PCI	LOCCFG_UNLOCK	Local configuration registers access control, see *Table 96 on page 238*	0x00034

Table 149: Register address list

Block	Register name	Description	Address offset
PCI	LSR[0]	PCI local space register 0, see *Table 78 on page 202*	0x00014
PCI	MAXLAT	PCI maximum latency, see *Table 120 on page 265*	0x1003F
PCI	MBAR[0]	PCI memory base address 0, see *Table 112 on page 259*	0x10010
PCI	MBMR	PCI memory space bank mask register, see *Table 92 on page 233*	0x00070
PCI	MBR	PCI memory space bank register, see *Table 88 on page 229*	0x00050
PCI	MINGNT	PCI minimum grant, see *Table 119 on page 265*	0x1003E
PCI	MLT	PCI latency timer, see *Table 109 on page 256*	0x1000D
PCI	NIP	PCI next item pointer, see *Table 124 on page 267*	0x100DD
PCI	PAR	PCI PIO address register, see *Table 87 on page 226*	0x0004C
PCI	PCDD	PCI power consumption/ dissipation data, see *Table 128 on page 277*	0x100E3
PCI	PDR	PCI PIO data register, see *Table 94 on page 236*	0x00078
PCI	PERF[N]	Performance registers These are implementation specific. See the datasheet for details.	0x00080 to 0x0008C
PCI	PINT	PCI power management interrupt register, see *Table 90 on page 231*	0x00058
PCI	PINTM	PCI power management interrupt mask register, see *Table 91 on page 232*	0x0005C

Table 149: Register address list

Block	Register name	Description	Address offset
PCI	PMC	PCI power management capability, see *Table 125 on page 268*	0x100DE
PCI	PMCSR	PCI power management control and status, see *Table 126 on page 271*	0x100E0
PCI	PMCSR_BSE	PCI PMCSR bridge support extension, see *Table 127 on page 274*	0x100E2
PCI	RBAR0	Region 0 base address register, see *Table 97 on page 239*	0x00100
PCI	RBAR1	Region 1 base address register, see *Table 97 on page 239*	0x00110
PCI	RBAR2	Region 2 base address register, see *Table 97 on page 239*	0x00120
PCI	RBAR3	Region 3 base address register, see *Table 97 on page 239*	0x00130
PCI	RBAR4	Region 4 base address register, see *Table 97 on page 239*	0x00140
PCI	RBAR5	Region 5 base address register, see *Table 97 on page 239*	0x00150
PCI	RBAR6	Region 6 base address register, see *Table 97 on page 239*	0x00160
PCI	RBAR7	Region 7 base address register, see *Table 97 on page 239*	0x00170
PCI	RETRYTIME	PCI maximum latency, see *Table 122 on page 266*	0x10041
PCI	RID	PCI revision ID, see *Table 106 on page 255*	0x10008
PCI	RLAR0	Region 0 local address register, see *Table 100 on page 243*	0x00108
PCI	RLAR1	Region 1 local address register, see *Table 100 on page 243*	0x00118

Table 149: Register address list

Block	Register name	Description	Address offset
PCI	RLAR2	Region 2 local address register, see *Table 100 on page 243*	0x00128
PCI	RLAR3	Region 3 local address register, see *Table 100 on page 243*	0x00138
PCI	RLAR4	Region 4 local address register, see *Table 100 on page 243*	0x00148
PCI	RLAR5	Region 5 local address register, see *Table 100 on page 243*	0x00158
PCI	RLAR6	Region 6 local address register, see *Table 100 on page 243*	0x00168
PCI	RLAR7	Region 7 local address register, see *Table 100 on page 243*	0x00178
PCI	RSR0	Region 0 space register, see *Table 98 on page 240*	0x00104
PCI	RSR1	Region 1 space register, see *Table 98 on page 240*	0x00114
PCI	RSR2	Region 2 space register, see *Table 98 on page 240*	0x00124
PCI	RSR3	Region 3 space register, see *Table 98 on page 240*	0x00134
PCI	RSR4	Region 4 space register, see *Table 98 on page 240*	0x00144
PCI	RSR5	Region 5 space register, see *Table 98 on page 240*	0x00154
PCI	RSR6	Region 6 space register, see *Table 98 on page 240*	0x00164
PCI	RSR7	Region 7 space register, see *Table 98 on page 240*	0x00174
PCI	SID	PCI subsystem ID, see *Table 115 on page 263*	0x1002E

Table 149: Register address list

Block	Register name	Description	Address offset
PCI	STATUS	PCI status, see *Table 105 on page 250*	0x10006
PCI	SVID	PCI subsystem vendor ID, see *Table 114 on page 262*	0x1002C
PCI	TRDYTIME	PCI TRDY time-out, see *Table 121 on page 266*	0x10040
PCI	VCR.STATUS	Version control register status, see *Table 75 on page 192*	0x00000
PCI	VCR.VERSION	Version control register version, see *Table 76 on page 194*	0x00008
PCI	VID	PCI vendor ID, see *Table 102 on page 245*	0x10000
PCI	WCBAR	Local configuration registers base address, see *Table 95 on page 237*	0x0007C

Table 149: Register address list

Index