MultiPort Memory Controller (GX176)

Revision: r0p1

Technical Reference Manual

ARM DDI 0278B

MultiPort Memory Controller (GX176)
Technical Reference Manual

Release Information

The following changes have been made to this document.

Change history

Date	Issue	Change
28 August 2003	A	First release
17 October 2003	B	Update for r0p1

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
MultiPort Memory Controller (GX176) Technical Reference Manual

List of Tables
MultiPort Memory Controller (GX176) Technical Reference Manual

List of Figures
MultiPort Memory Controller (GX176) Technical Reference Manual

Preface

This preface introduces the *ARM MultiPort Memory Controller (GX176) Revision: r0p1 Technical Reference Manual*. It contains the following sections:

- *About this document* on page xiv
- *Feedback* on page xviii.

About this document

This document is the technical reference manual for the ARM *MultiPort Memory Controller* (MPMC).

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this document, where:

r*n*	Identifies the major revision of the product.
p*n*	Identifies the minor revision or modification status of the product.

Intended audience

This document has been written for hardware and software engineers implementing *System-on-Chip* (SOC) designs. It provides information to enable designers to integrate the peripheral into a target system as quickly as possible.

Using this manual

This document is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the ARM MPMC (GX176).

Chapter 2 *Functional Overview*

Read this chapter for a description of the major functional blocks of the ARM MPMC (GX176).

Chapter 3 *Programmer's Model*

Read this chapter for a description of the ARM MPMC (GX176) registers and programming details.

Chapter 4 *Programmer's Model for Test*

Read this chapter for a description of the logic in the ARM MPMC (GX176) for functional verification and production testing.

Appendix A *Signal Descriptions*

Read this appendix for details of the ARM MPMC (GX176) signals.

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams*
- *Signals* on page xvi
- *Numbering* on page xvii.

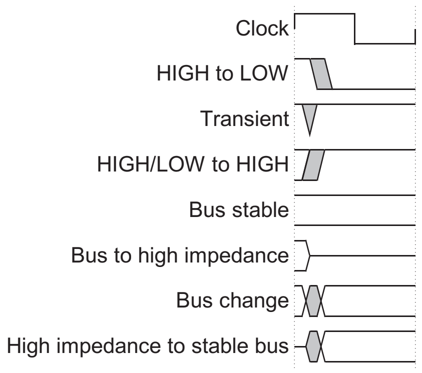
Typographical

The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace` (mono underlined)	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example: • `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>` • The Opcode_2 value selects which register is accessed.

Timing diagrams

The figure named *Key to timing diagram conventions* on page xvi explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level	The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means HIGH for active-HIGH signals and LOW for active-LOW signals.
Prefix A	Denotes *Advanced eXtensible Interface* (AXI) global and address channel signals.
Prefix B	Denotes AXI write response channel signals.
Prefix C	Denotes AXI low-power interface signals.
Prefix H	Denotes *Advanced High-performance Bus* (AHB) signals.
Prefix n	Denotes active-LOW signals except in the case of AHB or *Advanced Peripheral Bus* (APB) reset signals.
Prefix P	Denotes APB signals.
Prefix R	Denotes AXI read channel signals.
Prefix W	Denotes AXI write channel signals.
Suffix n	AHB **HRESETn** and APB **PRESETn** reset signals.

Numbering

The numbering convention is:

<size in bits>'<base><number>

This is a Verilog method of abbreviating constant numbers. For example:

- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.
- 8'h3F is an eight-bit wide hexadecimal value of `0x3F`. This is equivalent to b00111111.
- 8'b1111 is an eight-bit wide binary value of b00001111.

Further reading

This section lists publications by ARM Limited.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and the ARM Frequently Asked Questions list.

ARM publications

This document contains information that is specific to the ARM MPMC (GX176). Refer to the following documents for other relevant information:

- *AMBA Specification (Rev 2.0)* (ARM IHI 0011)
- *Systems IP ARM11 AMBA (Rev 2.0) AHB Extensions v1.0 Specification* (ARM IHI 0023)
- *AMBA Design Kit Technical Reference Manual* (ARM DDI 0243)
- *MBX R-S 3D Graphics Core Technical Reference Manual* (ARM DDI 0295)
- *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual* (ARM DDI 0249).

Feedback

ARM Limited welcomes feedback both on the ARM MPMC (GX176), and on the documentation.

Feedback on the ARM MPMC (GX176)

If you have any comments or suggestions about this product, contact your supplier giving:

- the product name
- a concise explanation of your comments.

Feedback on this document

If you have any comments on about this document, send an email to `errata@arm.com` giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- a concise explanation of your comments.

ARM Limited also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the ARM MPMC (GX176). It contains the following sections:

1.1 About the ARM MPMC (GX176)

The MPMC is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM Limited. It connects to the *Advanced High-performance Bus* (AHB) and a single dedicated MBX memory interface.

1.1.1 Features of the MPMC

The MPMC offers:

- AMBA 64-bit AHB support, including ARM11 AMBA extensions.
- Dedicated MBX Interface Port for direct connection to the ARM range of MBX 3D Graphics Cores.
- A separate AHB interface for programming the MPMC registers. Enables the MPMC registers to be situated in memory with other system peripheral registers.
- Locked AHB transactions supported.
- Support for all AHB burst types.
- Four chip selects for dynamic memory and four chip selects for static memory devices.
- Dynamic memory interface supports SDRAM, DDR-SDRAM, and low-power variants. It also supports Micron SyncFlash types of memory.
- Asynchronous static memory device support including RAM, ROM, Flash, and NAND Flash, with or without asynchronous page mode.
- 16-bit, 32-bit, and 64-bit wide databus SDRAM and SyncFlash memory support. 16-bit and 32-bit wide databus DDR-SDRAM support.
- 8-bit, 16-bit, and 32-bit wide static memory support.
- Static memory features include:
  - — asynchronous page mode read
  - — programmable wait states
  - — bus turnaround cycles
  - — output enable, and write enable delays
  - — extended wait.
- Read and write buffers to reduce latency and to improve performance.

- Controller supports 2K, 4K, and 8K row address synchronous memory parts. That is, typical 512Mb, 256Mb, 128Mb, and 16Mb parts, with 8, 16, 32, or 64 DQ (data) bits per device.
- Dynamic memory self-refresh mode supported by a *Power Management Unit* (PMU) interface or by software.
- Power-saving modes dynamically control **MPMCCKEOUT** and **MPMCCLKOUT**.
- Two reset domains enable dynamic memory contents to be preserved over a soft reset.
- Little, big, and mixed-endian support.
- Specifically designed for cached processors.
- Designed to work with noncritical word first, and critical word first processors, such as the ARM926EJ-S.
- Support for the *External Bus Interface* (EBI) that enables the memory controller pads to be shared.
- Integrated *Test Interface Controller* (TIC).
- PrimeCell ID support.

Note

Synchronous static memory devices (burst mode devices) are not supported.

1.2 Supported dynamic memory devices

This section provides examples of dynamic memory devices that are supported by the MPMC:

- *DDR-SDRAM devices*
- *SDRAM devices*
- *Micron style synchronous flash devices* on page 1-5
- *Micron style V-synchronous flash devices* on page 1-5
- *JEDEC low-power SDRAM devices* on page 1-5.

——— **Note** ———

This is not an exhaustive list of supported devices.

———————————

1.2.1 DDR-SDRAM devices

The MPMC supports the following DDR-SDRAM devices:

- 64Mb devices:
  - Micron MT46V2M32.
- 128Mb devices:
  - Micron MT46V16M8
  - Micron MT46V8M16.
- 256Mb devices:
  - Micron MT46V32M8.
  - Micron MT46V16M16.

1.2.2 SDRAM devices

The MPMC supports the following SDRAM devices:

- 16Mb devices:
  - Micron MT48LC1M16A1S
  - Samsung K4S160822D-G/F
  - Samsung K4S641632C.
- 64Mb devices:
  - Micron MT48LC2M32B2-6
  - Micron MT28S4M162C-10

  - Micron MT48LC4M16A2
  - Elpida µPD4564323-10
  - Hitachi HM5264805F-75.
- 128Mb devices:
  - Micron MT48LC4M32B2
  - Micron MT48LC16M8A2
  - Micron MT48LC8M16A2.
- 256Mb devices:
  - Elpida µPD45256163-10
  - Micron MT48LC16M16A2-8E
  - Hitachi HM522532F-B6
  - Hitachi HM5225805B-75.
- 512Mb devices:
  - Elpida HM5257805B-A6.

1.2.3 Micron style synchronous flash devices

The MPMC supports the following 64Mb devices:

- Micron MT28S4M16LC-10
- Micron MT28S4M16LC-12.

1.2.4 Micron style V-synchronous flash devices

The MPMC supports the 128Mb Micron MT28V4M32L.

1.2.5 JEDEC low-power SDRAM devices

The following JEDEC low-power SDRAM devices are supported:

- 64Mb Micron MT48LC2M32LFFC-8
- 128Mb Infineon HYB25L128160AC.

1.3 Supported static memory devices

This section provides examples of static memory devices that are supported by the MPMC:

- *Examples of ROM devices*
- *Examples of page mode ROM devices*
- *Examples of SRAM devices*
- *Examples of flash devices*
- *Examples of page mode flash devices*
- *Examples of NAND flash memory devices* on page 1-7.

——— **Note** ———

This is not an exhaustive list of supported devices.

———

1.3.1 Examples of ROM devices

The MPMC supports the 128Mb Samsung K3N9V(U)1000M-YC.

1.3.2 Examples of page mode ROM devices

The MPMC supports the 128Mb Samsung K3P9V(U)1000M-YC.

1.3.3 Examples of SRAM devices

The MPMC supports the following SRAM devices:

- 256Kb IDT IDT71V256SA20Y
- 256Kb Micron MT5C2568-12
- 4Mb Samsung K6F8016R6M
- 4Mb Samsung K6R4016CK-12
- 8Mb Samsung K6T8016C3M-70
- 8Mb Samsung K6F8008R2M.

1.3.4 Examples of flash devices

The MPMC supports the 4Mb Micron MT28F004B5.

1.3.5 Examples of page mode flash devices

The MPMC supports the 8Mb Intel 28F800F3 and the 4Mb Intel E28F320J3A110.

1.3.6 Examples of NAND flash memory devices

The MPMC supports the following NAND flash memory devices:

- 4Mb Samsung K9F4008W0A
- 64Mb AMD Am30LV0064D
- 256Mb Samsung K9F5608U08-Y
- 1Gb Toshiba TH58100FT
- 2Gb Samsung K9K2G08Q0M.

1.4 Product revisions

This section describes differences in functionality between product revisions of the MPMC (GX176):

r0p0-r0p1 Contains the following differences in functionality:

- Exclusive transfer state machine updated to match the *Systems IP ARM11 AMBA (Rev 2.0) AHB Extensions v1.0 Specification*.
- The 32-bit AHB ports and the 64-bit GXI port have been moved from exclusive transfer domain 0 to a domain not tracked by the four exclusive transfer state machines.

The above changes have no effect on the information provided in this manual.

Chapter 2
Functional Overview

This chapter describes the major functional blocks of the ARM MPMC (GX176). It contains the following sections:

2.1 MPMC functional description

Figure 2-1 shows a block diagram of the MPMC.

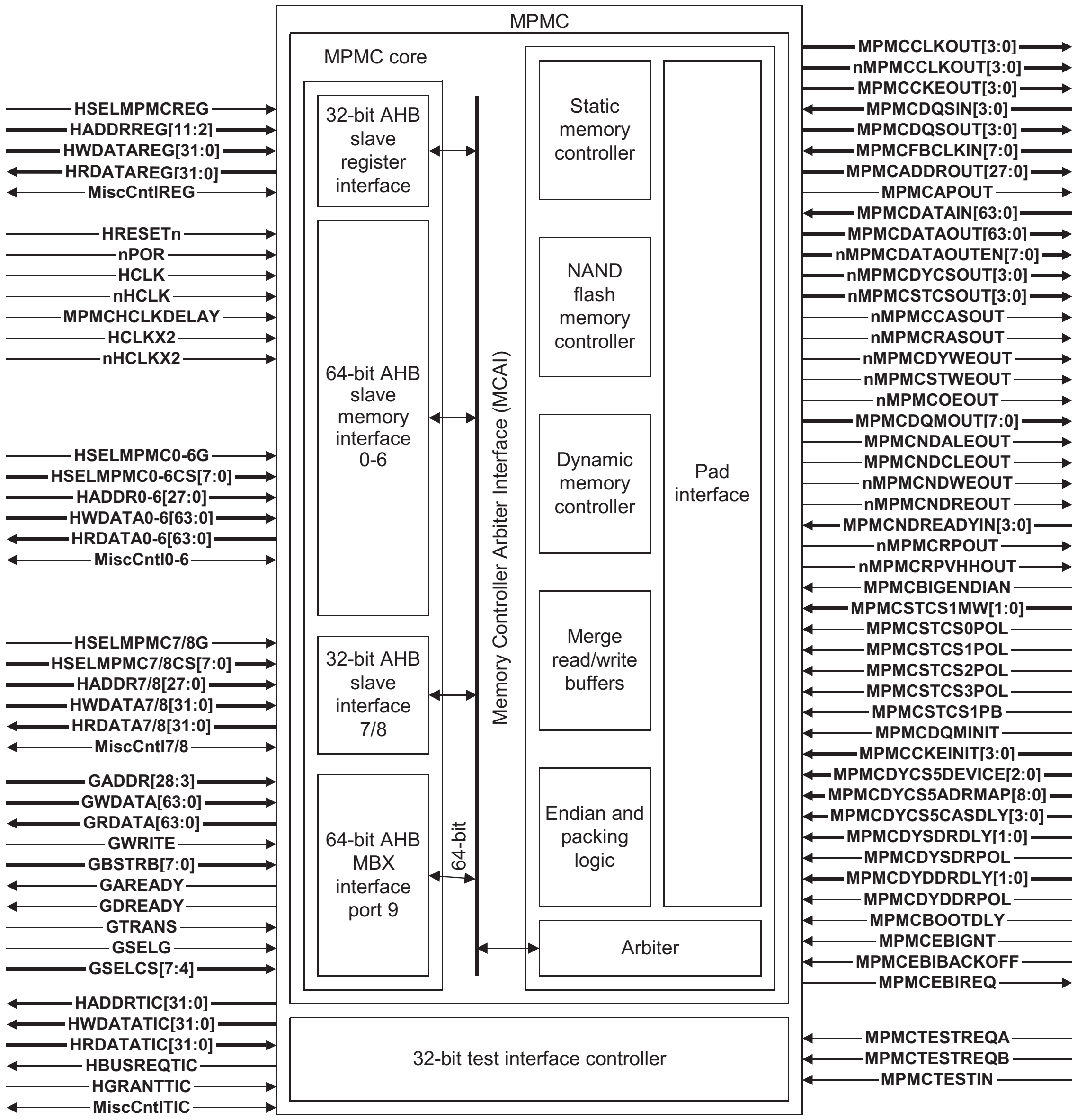


Figure 2-1 MPMC block diagram

The multiport memory controller block optimizes and controls external memory transactions. The functions of the MPMC blocks are described in the following sections:

- *Multiport memory controller block*
- *AHB slave register interface* on page 2-4
- *AHB slave memory interfaces* on page 2-5
- *MBX Interface Port* on page 2-10
- *Data buffers* on page 2-20
- *Endian and packing logic* on page 2-31
- *Arbiter* on page 2-32
- *Memory controller state machine* on page 2-32
- *Pad interface* on page 2-32
- *Test Interface Controller (TIC)* on page 2-33.

2.1.1 Multiport memory controller block

The multiport memory controller block optimizes and controls external memory transactions. The block contains the following:

- *Command sequencer*
- *Memory transfer state machine* on page 2-4
- *Static memory controller register bank* on page 2-4
- *Dynamic memory controller register bank* on page 2-4.

Command sequencer

The command sequencer holds up to 10 requests in its internal buffer. It prioritizes and rearranges accesses to maximize memory bandwidth and minimize transaction latency.

For example, if AHB interfaces 3 and 2 simultaneously request a data transfer from dynamic memory, to different memory banks, and the port 3 request address is to a closed page, but port 2 address is for an already open page, the following sequence occurs:

1. The ACT command is sent to open the SDRAM row specified by the AHB interface 3 address.
2. The AHB interface 2 access is completed.
3. AHB interface 3 access is completed.

The access priority is modified to take into account the ease of getting data to complete each transfer, but the access priority is always biased to the highest priority AHB interface.

Memory transfer state machine

The memory transfer state machine controls memory transactions.

Static memory controller register bank

There are four static memory controller register banks. Each contains the registers for a static memory bank.

Dynamic memory controller register bank

There are four dynamic memory controller register banks. Each contains the registers for a dynamic memory bank.

Note

The dynamic memory controller supports only SINGLE access bursts, that is:

- single transfer bursts for 64-bit wide external memory
- bursts of two transfers for 32-bit wide external memory
- bursts of four transfers for 16-bit wide external memory.

2.1.2 AHB slave register interface

The AHB slave register interface block enables the registers of the MPMC to be programmed. This module also contains most of the registers and performs most of the register address decoding. The register bank uses a 32-bit wide AHB interface. To connect a 64-bit wide AHB master to the register interface of the MPMC, an external downsizer, similar to that shown in Figure 2-2 on page 2-5, is required. A suitable downsizer is available as part of the *AMBA Design Kit* (ADK).

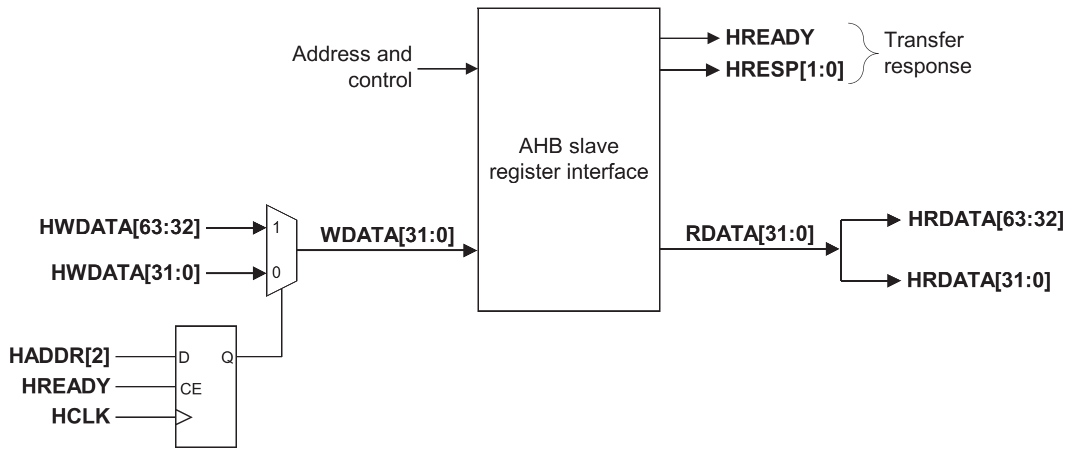


Figure 2-2 64-bit master to 32-bit slave register interface downsizer

The AHB slave register interface must be connected to the ARM processor AHB bus to enable the MPMC to be programmed.

AHB slave register transaction endianness and transfer width

To eliminate the possibility of endianness problems, all data transfers to and from the registers of the MPMC must be 32 bits wide.

Note

If an access is attempted with a size other than a word, 32 bits, it causes an ERROR response on **HRESP[0]** and the transfer is terminated.

2.1.3 AHB slave memory interfaces

The AHB slave memory interfaces enable devices to access the external memories. The memory interfaces are prioritized, with interface 0 having the highest priority. Having more than one memory interface enables high-bandwidth peripherals direct access to the MPMC, without data having to pass over the main system bus. This reduces memory access latency.

Note

- All AHB burst types are supported, enabling the most efficient use of memory bandwidth.

- The AHB interfaces do not generate SPLIT and RETRY responses.

AHB masters cannot be made to function on a narrower bus than originally intended, unless there is some mechanism included within the master to limit the width of transfers that the bus master attempts. The MPMC is designed to provide a selected mix of 32-bit and 64-bit AHB interfaces on a port-by-port basis. The required port mix can be configured as required for the application, either by adding or removing 32-bit or 64-bit ports, or by changing the order and priority of the port instantiations. If it is necessary to connect a 32-bit AHB master to one of the 64-bit AHB memory slave interfaces, an external conditioning block, similar to that shown in Figure 2-3, is required.

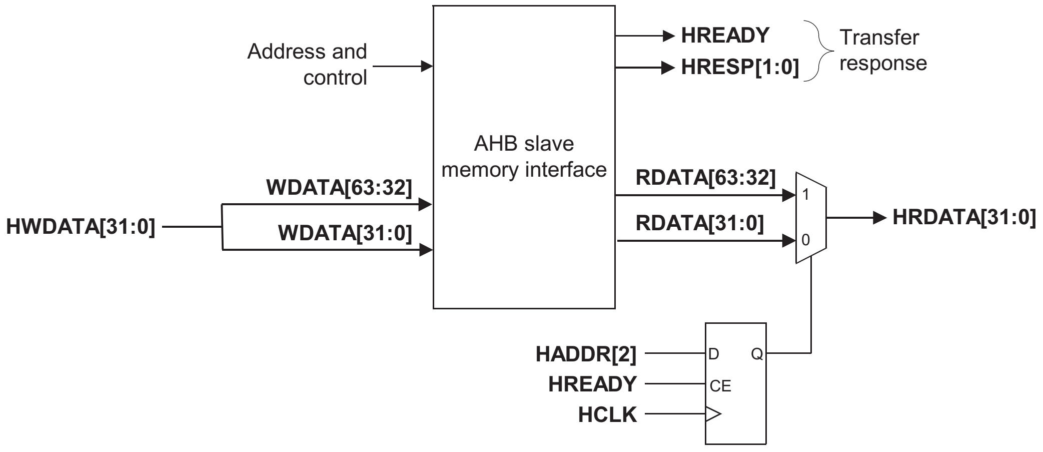


Figure 2-3 32-bit master to 64-bit slave memory interface

Note

If a 32-bit master is required on a high priority port, it is more efficient to replace a 64-bit port with a 32-bit port, rather than adding external logic to enable the 32-bit master to interface to a 64-bit port.

Memory transaction endianness

The following rules are applicable when handling memory transactions in the MPMC:

32-bit AHB ports

If the global **MPMCBIGENDIAN** signal is LOW, all 32-bit AHB ports are fixed as little-endian.

If the global **MPMCBIGENDIAN** signal is HIGH, all 32-bit AHB ports are fixed as big-endian.

64-bit AHB ports

If the global **MPMCBIGENDIAN** signal is LOW and the **MPMCBSTRBENx** signal of a port is LOW, that port is fixed as little-endian.

If the global **MPMCBIGENDIAN** signal is LOW and the **MPMCBSTRBENx** signal of a port is HIGH, that port is fixed as mixed-endian.

If the global **MPMCBIGENDIAN** signal is HIGH, all ports are fixed as big-endian irrespective of the state of their **MPMCBSTRBENx** signal.

External memory banks

If the global **MPMCBIGENDIAN** signal is LOW, all external memory banks are fixed as little-endian.

If the global **MPMCBIGENDIAN** signal is HIGH, all external memory banks are fixed as big-endian.

Table 2-1 shows a truth table for the MPMC endianness conditions.

Table 2-1 MPMC endianness options

MPMCBIGENDIAN	MPMCBSTRBENx	64-bit AHB port	32-bit AHB port	External memory bank
0	0	Little-endian	Little-endian	Little-endian
0	1	Mixed-endian	Little-endian	Little-endian
1	X (don’t care)	ARM big-endian	ARM big-endian	ARM big-endian

Note

MPMCBSTRBENx is set HIGH to enable byte strobes on a port, and to indicate that a port is compliant with ARMv6 masters. The master provides byte strobes on the **HBSTRB[7:0]** lines. When set HIGH, the byte strobe signals are always used by the MPMC and therefore must be driven to the correct values on all transfers, regardless of the **HUNALIGN** value. The ARM11 drives the byte strobes in this way.

MPMCBSTRBENx is set LOW to indicate that a port is compliant with ARMv5 and earlier masters only. The endianness of the data transfers to and from the external memories is then ARMv5 little-endian or ARMv5 big-endian, determined by the state of the global endian mode signal, **MPMCBIGENDIAN**.

The power-on reset value of the global **MPMCBIGENDIAN** signal can be overridden through the register interface by accessing the MPMCConfig Register.

The setting of ARM11 mixed-endian mode for a port cannot be overridden through the register interface. It is fixed at power-on reset.

All data in the MPMC must be flushed when switching between little-endian and big-endian modes. This is so that data residing in the merge buffers is transferred correctly.

Memory transaction size

Memory transactions can be 8, 16, 32, or 64 bits wide. Any access attempted with a size greater than the width of the AHB memory ports causes an ERROR response on **HRESP[0]** and the transfer is terminated.

Unpopulated memory areas

Accesses to address space within a memory bank which is not populated generates an ERROR response on **HRESP[0]** and the transfer is terminated. The system decoder defines the populated address space within a memory bank using a full decode of the corresponding **HSELMPMCxCSy** select signal.

The MPMC is not able to detect unpopulated memory areas, so the system decoder must fully define the memory map and ensure that the **HSELMPMCxCSy** signals are only driven when populated memory locations are addressed.

Write protected memory areas

Write transactions to write-protected memory areas generate an ERROR response on **HRESP[0]** and the transfer is terminated.

Non-coherency

The data ports are non-coherent, because there is no central buffer or transfer checking mechanism in the MPMC. This means that reads and writes to the same location in memory are only coherent when the data is available in the memory device.

For example, if a write is pending on a low priority port and a read from the same location is performed from a higher priority port which is granted before the write, the value returned is the data read from that address in memory, not the value pending to be written by the other port.

Devices which require coherency over different data ports must use a semaphore mechanism to indicate when data is available.

For an AHB master such as the ARM11 which has separate read and write data ports which much be coherent, disabling the AHB port buffer inserts wait states after a write transfer until the write is performed to memory. This ensures that a read to the same address following the write returns the value written to memory.

ARM11 AMBA extensions

64-bit AHB ports provide support for the ARM11 AMBA extensions. This includes the following:

- exclusive accesses using extended **HPROT** and **HRESP** lines
- mixed-endian transfers using the **HBSTRB** byte strobe signals
- unaligned transfers using the byte strobe and unalign signals:
  - **HBSTRB**
  - **HUNALIGN**.

For further information, see the *Systems IP ARM11 AMBA (Rev 2.0) AHB Extensions v1.0 Specification*.

Exclusive transfer monitors

Exclusive transfers use the **HDOMAIN** input to determine if transfers from different AHB ports are from the same master, for example, ARM11 read and write data ports. The 64-bit GXI port and 32-bit AHB ports are in a separate domain, because only 64-bit AHB ports can perform exclusive transfers.

Four exclusive transfer monitors are provided, with each monitor dedicated to a single domain. This enables four exclusive accesses from four different domains to four different addresses to be in progress at any one time.

2.1.4 MBX Interface Port

The MBX Interface Port is described in:

- *Slave select signal generation*
- *Transaction descriptions*
- *Typical MBX 3D Graphics Core data transfer profiles* on page 2-19.

Slave select signal generation

The MBX interface port can only access dynamic memory devices using input signals **GSELCS[7:4]**. Static memory cannot be accessed by the MBX interface port. The **GSELG** and **GSELCS[7:4]** signals must be generated by a decoder between the MBX 3D Graphics Core and the MPMC. This is similar to an AHB decoder, where the upper bits of the current address are decoded to select the MPMC and the chip select to use for the memory transfer.

If there are multiple slaves connected to the MBX 3D Graphics Core, the above decoding is required for the **GSELG** signal. However, if the MPMC is the only slave, the **GTRANS** signal can be used instead because every valid transfer selects the MPMC.

If the MBX 3D Graphics Core accesses memory on more than one chip select, the above decoding is required for the **GSELCS[7:4]** signals. However, if only one chip select is used, this bit of **GSELCS** can be driven in the same way as **GSELG**, with the other **GSELCS** bits tied LOW.

Transaction descriptions

Data is transferred on any cycle when **GTRANS** and **GAREADY** are both HIGH on a rising **HCLK** edge. MBX 3D Graphics Core transactions are described in the following sections:

- *Read transactions* on page 2-11
- *Write transactions* on page 2-14
- *Read/write transactions* on page 2-17
- *GBSTRB functionality* on page 2-17
- *Additional considerations* on page 2-19.

——— **Note** ———

The MBX only supports little-endian format transfers, and the operation of the MBX Interface Port is unaffected by the **MPMCBIGENDIAN** input or N bit of the MPMCConfig Register. Reads of data written by the MBX are always performed correctly.

However, the byte ordering of data can be changed in the following situations when the MPMC is configured to be in big-endian mode:

- the MBX reads data written by an AHB device in big-endian format
- an AHB device reads data written by the MBX in little-endian format.

Read transactions

Read transfers are split into two distinct phases:

- address phase
- data phase.

To start a read transfer, the MBX 3D Graphics Core memory interface:

- drives **GWRITE** LOW
- drives an address onto **GADDR**
- drives **GBSTRB**
- indicates that these signals are valid by driving **GTRANS** HIGH.

This effectively requests use of the memory, and the transfer is initiated when the MPMC drives **GAREADY** HIGH if it is not already HIGH. Read transfers then occur on every subsequent clock, unless **GAREADY** is taken LOW, effectively inserting wait states into the address phase.

Data is returned, in order, by the MPMC at a later time on the **GRDATA** bus. The data on this bus is valid when **GDREADY** is HIGH. This is a split protocol. The ordering of **GRDATA** responses is the same as **GADDR** responses, because there is no out-of-order return of data.

Figure 2-4 on page 2-12 shows a read of five words by the MBX 3D Graphics Core. The diagram shows the memory access as having a latency of eight clocks.

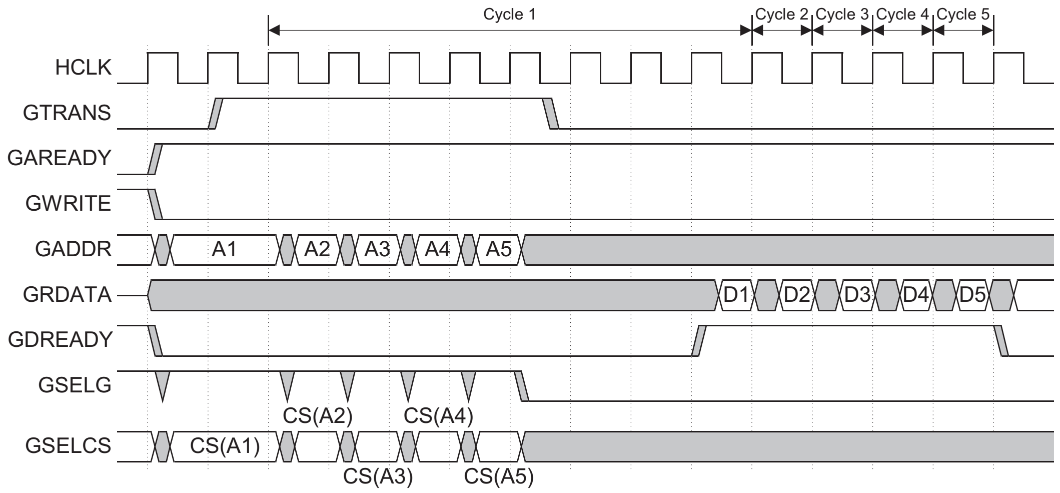


Figure 2-4 MBX 3D Graphics Core read, five transactions

Figure 2-5 on page 2-13 shows a large number of read transactions. The last three transactions are held off for two clocks by the MPMC by driving **GAREADY** LOW, for example, because of buffers in the MPMC filling. It also shows returned data being marked as invalid, with **GDREADY** being driven LOW. This adds more latency to the held-off transactions. This might be caused by the MPMC having to open a new memory page to access address A7 and return D7.

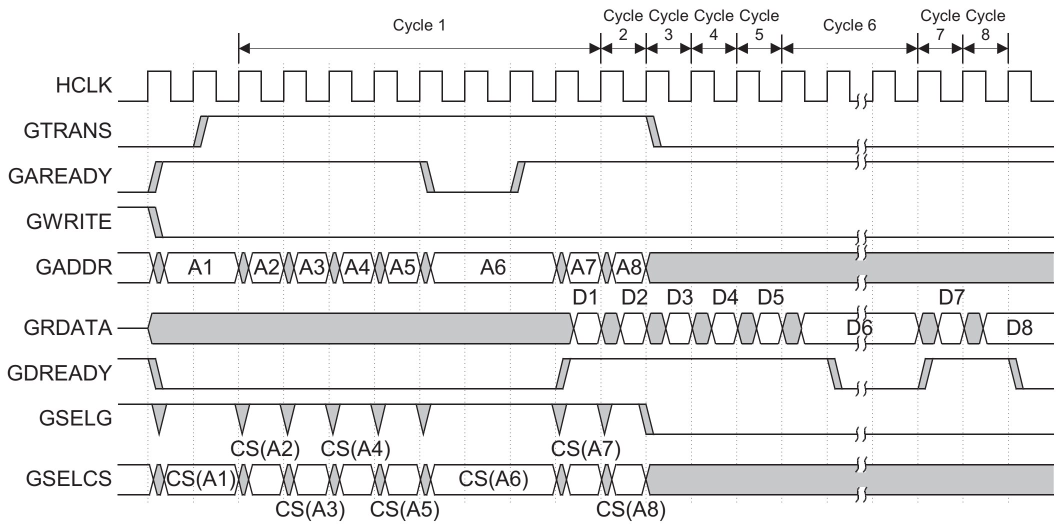


Figure 2-5 MBX 3D Graphics Core read, multiple transactions

Figure 2-6 shows the MBX 3D Graphics Core pausing its activity on the memory interface by driving **GTRANS** LOW. Addresses A4, A5, and A6 are not read.

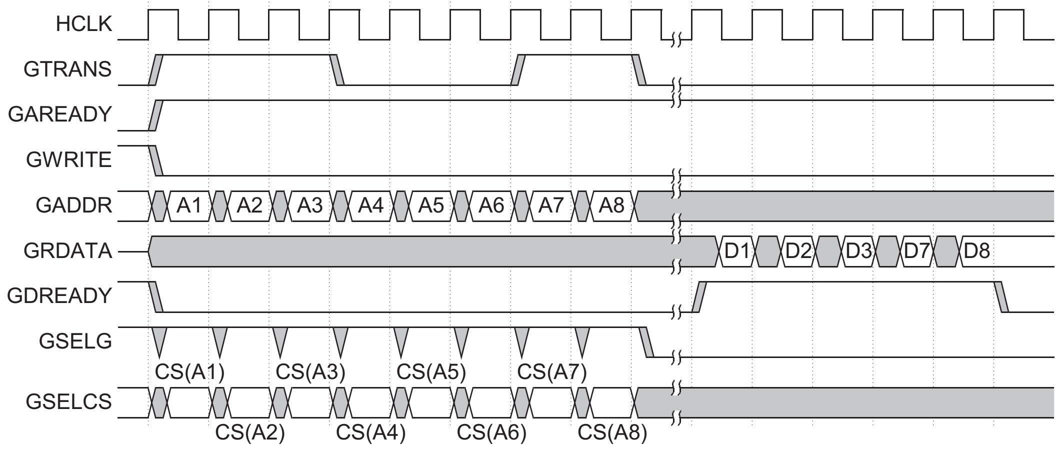


Figure 2-6 MBX 3D Graphics Core read, paused

Figure 2-7 shows a series of read request transactions on the MBX 3D Graphics Core memory interface. The address requests consist of five individual requests:

- the first is performed without delay
- the second is stalled by the MBX 3D Graphics Core for one clock cycle
- the third access is stalled by both the MPMC and the MBX 3D Graphics Core for three clock cycles because of the combined effects of the **GAREADY** and **GDREADY** signals
- the MPMC returns the read data with the fourth and fifth pieces of data being stalled for one clock cycle.

This also demonstrates that return data can happen concurrently with address request cycles.

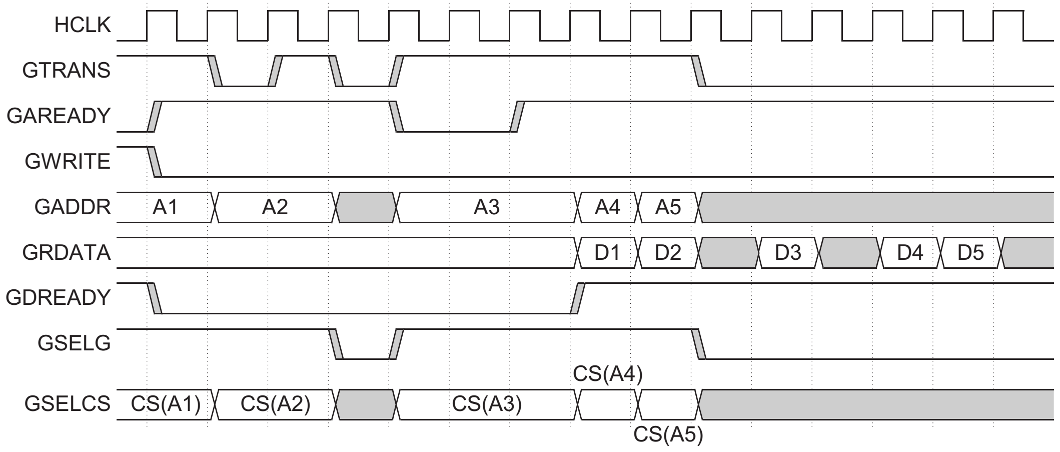


Figure 2-7 MBX 3D Graphics Core read, stalled

Write transactions

To start a write transfer, the MBX 3D Graphics Core memory interface:

- drives **GWRITE** HIGH
- drives **GADDR**, **GWDATA**, and **GBSTRB**
- signals that these signals are valid by driving **GTRANS** HIGH.

This effectively requests use of the memory, and the transfer is initiated when the MPMC drives **GAREADY** HIGH, if it is not already. Write transfers then occur on every subsequent clock, unless **GAREADY** is taken LOW, inserting wait states into the transfer.

Note

The value of **GDREADY** is ignored on writes.

Figure 2-8 shows a series of eight write transactions.

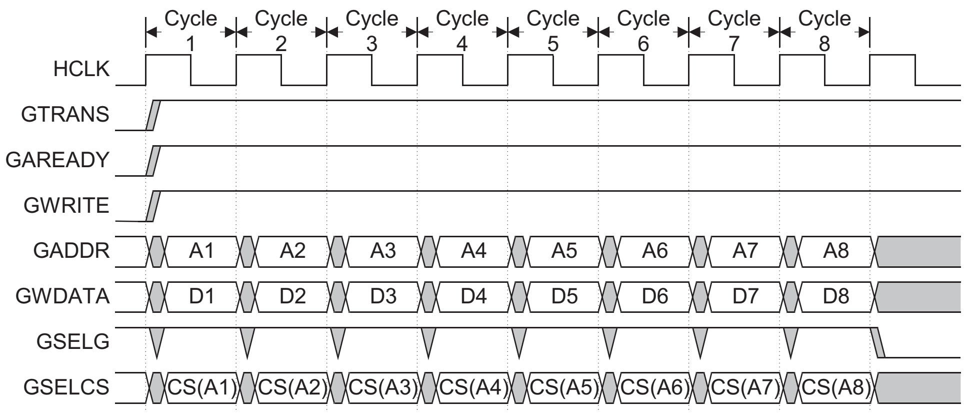


Figure 2-8 MBX 3D Graphics Core write, eight transactions

Figure 2-9 on page 2-16 shows five write cycles. The first and fifth cycles are extended by the memory controller, by driving **GAREADY** LOW.

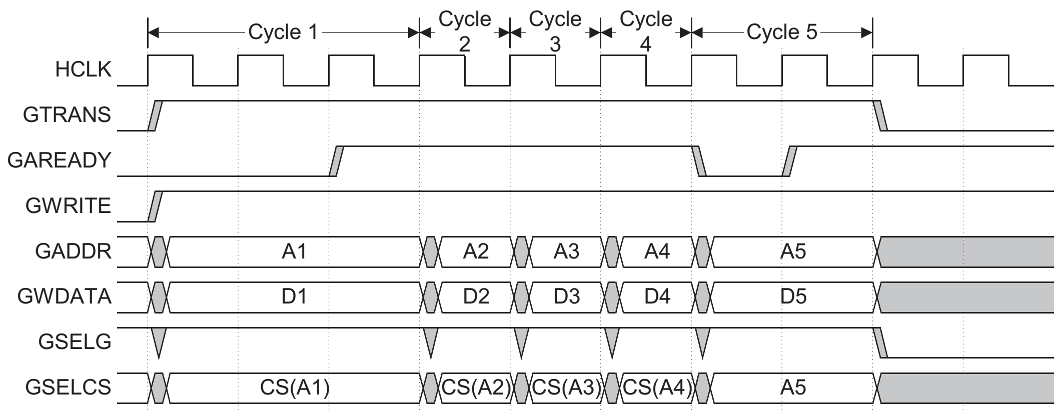


Figure 2-9 MBX 3D Graphics Core write, five transactions

Figure 2-10 shows six write cycles. **GTRANS** LOW indicates that the data on the address and data buses, **GADDR** and **GWDATA** respectively, are not valid and so no write transactions take place for those clock periods despite **GAREADY** being HIGH.

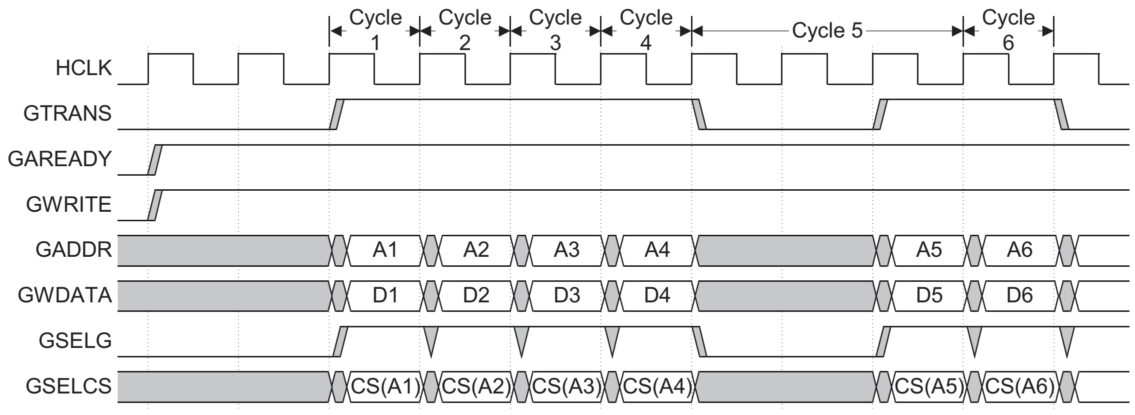


Figure 2-10 MBX 3D Graphics Core write, six transactions

Read/write transactions

Figure 2-11 shows a series of write request transactions on the MBX 3D Graphics Core memory interface with read request data being returned from previous read request transactions. This shows that read data can overlap with write requests, ensuring maximum bus efficiency.

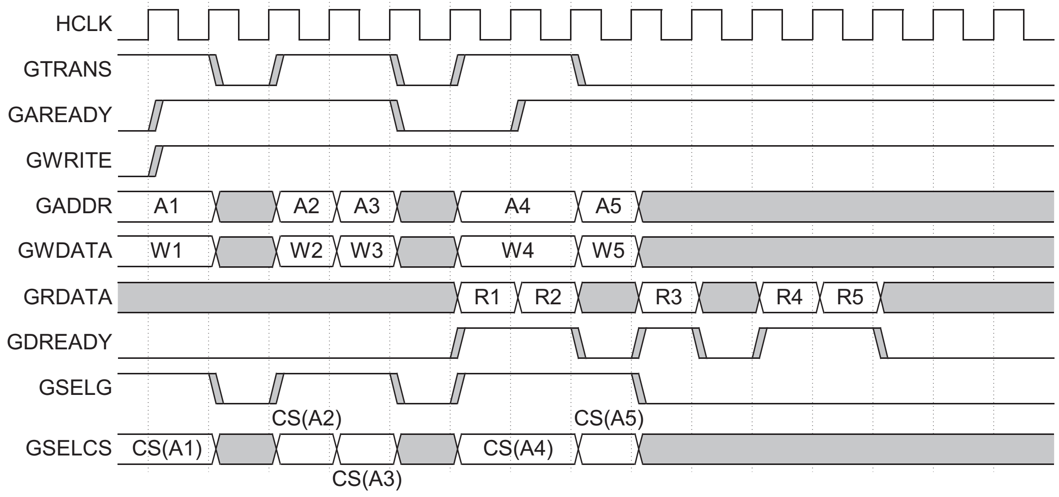


Figure 2-11 MBX 3D Graphics Core read/write transactions

GBSTRB functionality

The MBX interface uses the **GBSTRB** signals as byte lane strobes to enable subword writes.

GBSTRB is an eight-bit byte lane enable signal. A high on the relevant byte mask bit indicates that the byte is valid and must be written to SDRAM.

Table 2-2 shows which bits of **GBSTRB** relate to which byte lanes in **GWDATA**.

Table 2-2 GBSTRB bits

GBSTRB bit	GWDATA byte
[0]	[7:0]
[1]	[15:8]
[2]	[23:16]

Table 2-2 GBSTRB bits (continued)

GBSTRB bit	GWDATA byte
[3]	[31:24]
[4]	[39:32]
[5]	[47:40]
[6]	[55:48]
[7]	[63:56]

Figure 2-12 shows how **GBSTRB** functions. It shows three write transactions to SDRAM through the memory controller.

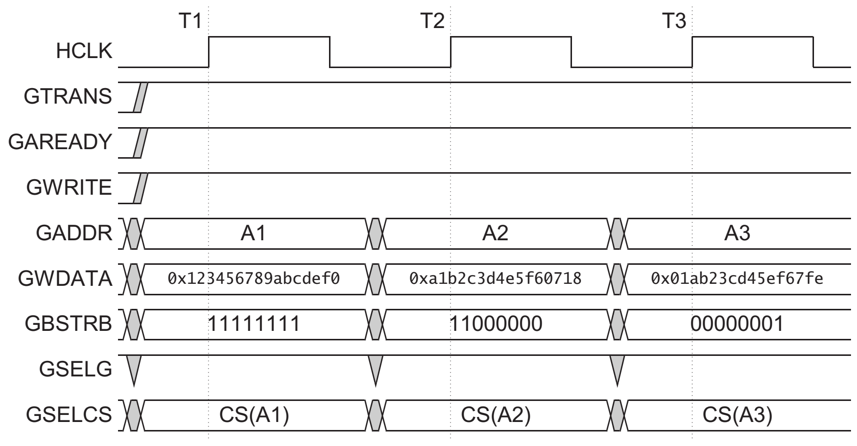


Figure 2-12 GBSTRB timing diagram

Table 2-3 shows the GBSTRB transactions.

Table 2-3 GBSTRB transactions

Transaction	Activity
T1	Writes the complete 64-bit data word to address A1.
T2	Writes `0xa1b2` to the top 16 bits of address A2. The bottom 24 bits remain unchanged.
T3	Writes `0xfe` to the bottom byte of address A3

Additional considerations

The MPMC only initiates a transaction when both **GTRANS** and **GAREADY** are HIGH.

Typical MBX 3D Graphics Core data transfer profiles

Memory accesses originating from the MBX 3D Graphics Core originate from a variety of internal sources. An example relative distribution of these accesses for the ARM MBX HR-S is shown in Table 2-4.

Note

Burst does not mean that the accesses are to contiguous addresses as is the case with AMBA. It implies that there is a group of requests originating from a given internal port, that are generally in the same memory page.

Table 2-4 Example ARM MBX HR-S memory transfer profile

Access port	Burst type	Read/ write	Access type	Scalable	Bandwidth	% of bandwidth
TARW	Usually burst	Read/ write	Random (usually in page)	Yes	17.01MB/s	18.28
PARAM (ISP pointer read)	Bursts of 4	Read	Random (usually in page)	Yes	2.23MB/s	2.40
PARAM (ISP vertex read)	Bursts of 12	Read	Random (usually in page)	Yes	27.3MB/s	29.33
PARAM (TSP vertex read)	Bursts of 20	Read	Random (usually in page)	Yes	26.38MB/s	28.34
ZLZS (Z load/Z store port)	256 word bursts	Read/ write	Random (usually in page)	Yes	0MB/s	0
TEX (texture cache port)	Bursts of 2 or 4	Read	Random (often not in page)	Yes	15.55MB/s	16.71
PIX (pixel write port)	256 word bursts	Write	Contiguous in page	No	4.61MB/s	4.95
Total bandwidth					93.08MB/s	100

Note

Because Z load/store is not normally enabled in this typical example case, Z load/store bandwidth is zero. The proportion of the bandwidth that each port requires depends on the input data. The figures in Table 2-4 on page 2-19 represent a profile for a system based on the following data:

- MBX HR-S core
- screen size of 320x240 pixels
- screen color depth of 16bpp
- frame rate of 30 *frames per second* (fps)
- input number of vertices is 20 000
- texture mode of 16bpp
- texture cache hit rate of 85%.

The MBX core arbitrates internally between the requestors.

The MBX Interface Port is the lowest priority port on the MPMC. MBX 3D Graphics Core accesses are high-bandwidth but are not latency-critical. MBX 3D Graphics Core are therefore only allocated what memory bandwidth is left over after all other devices have used what they require. This ensures that the MBX Interface Port cannot starve other devices of bandwidth.

2.1.5 Data buffers

Each AHB memory interface uses a single Dword write buffer and read cache to merge data to and from external memory to improve memory bandwidth. When reading word (32-bit), halfword, or byte wide data, external memory accesses are reduced to a single 64-bit wide external memory access. If consecutive address hits occur to data cached in the 64-bit wide read buffer, no external memory accesses are necessary. Similarly, when writing word, halfword, or byte data, external memory accesses are reduced to a single 64-bit wide access if consecutive address hits occur to the 64-bit wide write buffer. The merge write or read buffers for each port can be disabled using the register interface by writing to the MPMCAHBControlx Register for the AHB port.

Buffered write and read transactions on a single AHB port to the same memory location are always coherent.

Note

For 32 and 64-bit masters which use separate read and write ports, the associated AHB port buffers must be disabled to ensure that data coherency is maintained.

The internal merge buffers of both 32 and 64-bit ports are fixed at 64 bits, and any external read accesses always perform enough transfers to fill the 64-bit buffer, assuming they are enabled and **HPROT** indicates it can be cached or buffered.

The AHB port data buffers are only used for dynamic memory transfers. The static memory controller uses a 32-bit buffer to optimize reads and writes.

The buffers are automatically disabled during ARM11 Exclusive Access transactions.

Note

The MBX interface port does not contain a buffer similar to the AHB ports. All transfers through the MBX interface port are 64-bit with optional byte lane strobes and do not form parts of bursts, so a buffer is not required to maximize memory performance for bursts with a size of less than 64 bits.

Read transaction buffer operation

If an 8-bit, 16-bit, or 32-bit wide AHB read transfer is performed with the buffers enabled, the read transaction submitted to the memory is at the maximum width of the memory chip-select, with enough transfers to fill the 64-bit buffer. Therefore a chip-select with a 64-bit databus returns 64 bits of data. This data is then placed into the buffer. Data is provided from the buffer to the AHB bus as necessary. This reduces the number of read transactions to external memory for 8-bit, 16-bit, or 32-bit wide AHB transactions. The memory controller can then re-arbitrate to a different AHB port and perform memory transactions while the data is being transferred from the data buffer.

Note

- Enabling the buffers has no impact on 64-bit wide read transactions.
- The memory controller re-arbitrates to an open page transfer during a buffered transaction.

Enabling read buffer

For read transactions the respective AHB port buffer is only used if:

- The AHB respective AHB port buffer is enabled.
- One of the 8-bit, 16-bit, or 32-bit wide AHB read transactions are performed. 64-bit wide transactions do not use the buffer, because this does not improve performance.

- A multi-word AHB burst is performed. AHB burst SINGLE transfers do not use the buffer.
- AHB HPROT protection information indicates that the transfer is cacheable, indicating to the memory controller that the particular read transaction can be buffered.
- The transaction is not a locked transfer (**HMASTLOCK** is LOW). This ensures that atomic AHB transactions are performed straight to memory and are not re-arbitrated during the transaction.
- The transaction is not an ARM11 Exclusive Access read.

Note

If an AHB master does not provide AHB HPROT protection information, the relevant HPROT bits can be tied off as required.

Read data re-use

Only the AHB burst that fetched the data from the memory into the buffer can use the data in the buffer. Subsequent AHB bursts do not make use of the read data in the buffer even if the transaction is to the same memory area.

Advantages of read buffering

Enabling read buffering:

- reduces power consumption because fewer commands are issued to memory.
- increases memory bandwidth for 8-bit, 16-bit, and 32-bit wide memory transactions, because the memory controller can re-arbitrate to service a different AHB port while the data is being read from the buffer.

Note

Only AHB ports that have a memory request to open pages of SDRAM memory are serviced.

Read buffer example

The example described is for the case where the AHB port buffers are enabled and cacheable transfers are performed, with AHB port 0 and AHB port 1 performing INCR4, 16-bit wide read transactions. Accesses are to a 64-bit wide dynamic memory chip-select.

Read buffer enabled

Table 2-5 shows that with the read buffer enabled the memory is operating at maximum efficiency and providing 64 bits of data on each clock cycle.

Table 2-5 Read buffer enabled

Clock cycle	AHB 0	AHB 1
1	64-bit read from memory and placed in buffer. Data 0 returned on AHB.	AHB port waiting for data.
2	Data 1 returned on AHB.	64-bit read from memory and placed in buffer. Data 0 returned on AHB
3	64-bit read from memory and placed in buffer. Data 2 returned on AHB.	Data 1 returned on AHB.
4	Data 3 returned on AHB.	64-bit read from memory and placed in buffer. Data 2 returned on AHB.
5	AHB port available.	Data 3 returned on AHB.

Read buffer disabled

Table 2-6 shows that with the read buffer disabled the memory provides 32 bits of data on each clock cycle. The memory controller therefore only provides half the amount of bandwidth compared to the case with the buffers enabled.

Table 2-6 Read buffer disabled

Clock cycle	AHB 0	AHB 1
1	32-bit read from memory. Data 0 returned on AHB.	AHB port waiting for data.
2	32-bit read from memory. Data 1 returned on AHB.	AHB port waiting for data.
3	32-bit read from memory. Data 2 returned on AHB.	AHB port waiting for data.
4	32-bit read from memory. Data 3 returned on AHB.	AHB port waiting for data.
5	AHB port available.	32-bit read from memory. Data 0 returned on AHB.
6	AHB port available.	32-bit read from memory. Data 1 returned on AHB.
7	AHB port available.	32-bit read from memory. Data 2 returned on AHB.
8	AHB port available.	32-bit read from memory. Data 3 returned on AHB.

Disadvantages of read buffering

The disadvantage with enabling read buffering is that an 8-bit, 16-bit, or 32-bit wide read transfer can take longer to complete, because the AHB port is re-arbitrated to maximize bandwidth while the data is being read from the buffer. If buffering is not required, read buffering can be disabled by either:

- setting the Buffer enable (E) field of the MPMCAHBControl Register inactive
- setting the AHB HPROT protection information to indicate a non-cacheable access.

Worst case additional latency

The worst case additional latency is for the case where one AHB port, in this example AHB port 0, performs INCR16 32-bit wide read transactions. Other AHB ports, in this case AHB port 1 and 2, perform continuous, in page, INCR16 64-bit read transactions. Accesses are to a 32-bit wide dynamic memory chip-select.

Read buffer enabled

With the read buffer enabled the memory provides 64 bits of data on every clock cycle. The memory controller re-arbitrates to a different AHB port when data is being read from the buffer. The latency of a buffer transfer might therefore be longer than a nonbuffered transfer.

The worst case additional latency for a buffered transfer is when a buffered INCR16 32-bit wide burst is being performed and the other AHB ports are performing INCR16 64-bit wide transfers. Table 2-7 shows that the INCR16 32-bit wide read can take up to 128 cycles to complete.

Table 2-7 Read buffer enabled

Clock cycle	AHB 0	AHB 1	AHB 2
1	64-bit read from memory and placed in buffer. 16-bit data 0 returned on AHB.	AHB port waiting for data	AHB port waiting for data
2	32-bit data 1 returned on AHB.	AHB port waiting for data	AHB port waiting for data
3	AHB port waiting for data.	64-bit read zero from memory	AHB port waiting for data
4-17	AHB port waiting for data.	64-bit read 1-14 from memory	AHB port waiting for data
18	AHB port waiting for data.	64-bit read 15 from memory	AHB port waiting for data

Table 2-7 Read buffer enabled (continued)

Clock cycle	AHB 0	AHB 1	AHB 2
19	64-bit read from memory and placed in buffer. 32-bit data 2 returned on AHB.	Next INCR16 transfer	AHB port waiting for data
20	32-bit data 3 returned on AHB.	AHB port waiting for data	AHB port waiting for data
21	AHB port waiting for data.	AHB port waiting for data	64-bit read zero from memory
22-37	AHB port waiting for data.	AHB port waiting for data	64-bit read 1-14 from memory
38	AHB port waiting for data.	AHB port waiting for data	64-bit read 15 from memory
39	AHB port waiting for data.	AHB port waiting for data	Next INCR16 transfer
40	64-bit read from memory and placed in buffer. 32-bit data 4 returned on AHB.	AHB port waiting for data	AHB port waiting for data
41	32-bit data 5 returned on AHB.	AHB port waiting for data	AHB port waiting for data

Note

Table 2-7 on page 2-24 shows the first 41 cycles of the transaction. Cycles 42 to 136 are a continuation of the sequence shown.

Read buffer disabled

Table 2-8 shows that if read buffers are not enabled, an INCR16 32-bit wide read burst takes 16 cycles to complete when the read data starts being returned from the memory.

Table 2-8 Read buffer disabled

Clock cycle	AHB 0	AHB 1	AHB 2
1	32-bit read from memory. Data 0 returned on AHB.	AHB port waiting for data.	AHB port waiting for data
2-15	32-bit read from memory. Data 1-14 returned on AHB.	AHB port waiting for data.	AHB port waiting for data
16	32-bit read from memory. Data 15 returned on AHB.	AHB port waiting for data.	AHB port waiting for data

Table 2-8 Read buffer disabled (continued)

Clock cycle	AHB 0	AHB 1	AHB 2
17	AHB port available.	64-bit read from memory. Data 0 returned on AHB.	AHB port waiting for data
18-31	AHB port available.	64-bit read from memory. Data 1-14 returned on AHB.	AHB port waiting for data
32	AHB port available.	64-bit read from memory. Data 15 returned on AHB.	AHB port waiting for data3

Write transaction buffer operation

If an 8-bit, 16-bit, or 32-bit wide AHB write transfer is performed with the buffers enabled the write data is merged into the buffer, so that a write of the maximum width of the memory chip-select is performed. Therefore, for a chip-select with a 64-bit databus, 64 bits of data are written at a time. This reduces the number of write transactions to external memory for 8-bit, 16-bit, or 32-bit wide AHB transactions. The memory controller can then re-arbitrate to a different AHB port and perform memory transactions while the data is being written into the data buffer.

Note

- Enabling the buffers has no impact on 64-bit wide write transactions.
- The memory controller re-arbitrates to an open page transfer during a buffered transaction.

Enabling write buffer

For write transactions the respective AHB port buffer is only used if:

- The respective AHB port buffer is enabled.
- An 8-bit, 16-bit, or 32-bit write transaction is performed. 64-bit wide transactions do not make use of the buffer because this does not improve performance.
- A multi-word AHB burst is performed. AHB burst SINGLE transfers do not use the buffer.
- AHB HPROT protection information indicates that the transfer is bufferable. This indicates to the memory controller that the particular write transaction can be buffered.

- The transaction is not a locked transfer (**HMASTLOCK** is LOW). This ensures that atomic AHB transactions are performed straight to memory and are not re-arbitrated during the transaction.
- The transaction is not an ARM11 Exclusive Access write.

Note

If an AHB master does not provide AHB **HPROT** protection information the relevant **HPROT** bits can be tied off as required.

Write data re-use

If an AHB port performs a burst write into a buffer subsequent AHB burst writes do not merge data into the same buffer, even if the subsequent burst is to the same area of memory. Instead the data in the buffer from the first write is submitted to memory and only then can the second burst start. This ensures that the memory is updated at the end of each burst write so that if another AHB port reads from the same memory location the memory is updated.

Advantages of write buffering

Enabling write buffering:

- reduces power consumption as fewer commands are issued to memory
- increases memory bandwidth for 8-bit, 16-bit, and 32-bit wide memory transactions, because the memory controller can re-arbitrate to service a different AHB port while the data is being written to the buffer.

Note

Only AHB ports that have a memory request to open pages of SDRAM memory are serviced.

Write buffer example

The write buffer example describes the case where the AHB port buffers are enabled and bufferable transfers are performed, with AHB port 0 and AHB port 1 performing INCR4, 32-bit wide write transactions. Accesses are to a 64-bit wide dynamic memory chip-select.

Write buffer enabled

Table 2-9 shows that with the write buffer enabled the memory is operating at maximum efficiency and providing 64 bits of data on each clock cycle.

Table 2-9 Write buffer enabled

Clock cycle	AHB 0	AHB 1
1	32-bit write data 0 written into buffer.	AHB port waiting for data.
2	32-bit write data 1 written into buffer. 64-bit write data written to memory.	32-bit write data 0 written into buffer.
3	32-bit write data 2 written into buffer.	32-bit write data 1 written into buffer. 64-bit write data written to memory.
4	32-bit write data 3 written into buffer.	32-bit write data 2 written into buffer.
5	AHB port available.	32-bit write data 3 written into buffer. 64-bit write data written to memory.

Write buffer disabled

Table 2-10 shows that with the write buffer disabled 32 bits of data is written to memory on each clock cycle. The memory controller therefore only provides half the amount of bandwidth compared to the case with the buffers enabled.

Table 2-10 Write buffer disabled

Clock cycle	AHB 0	AHB 1
1	32-bit write data 0 written into memory	AHB port waiting for data
2	32-bit write data 1 written into memory	AHB port waiting for data
3	32-bit write data 2 written into memory	AHB port waiting for data
4	32-bit write data 3 written into memory	AHB port waiting for data
5	AHB port available	32-bit write data 0 written into memory
6	AHB port available	32-bit write data 1 written into memory
7	AHB port available	32-bit write data 2 written into memory
8	AHB port available	32-bit write data 3 written into memory

Disadvantages of write buffering

The disadvantage with enabling write buffering is that an 8-bit, 16-bit, or 32-bit wide write transfer can take longer to complete, because the AHB port is re-arbitrated to maximize bandwidth while the data is being written into the buffer. If buffering is not required, write buffering can be disabled by:

- setting the Buffer enable (E) field of the MPMCAHBControl Register inactive
- setting the AHB **HPROT** protection information to indicate a nonbufferable access.

Worst case additional latency

The worst case additional latency is for the case where one AHB port, in this case AHB port 0, performs INCR16 32-bit wide write transactions. Other AHB ports, in this case AHB port 1 and 2, perform continuous, in page, INCR16 64-bit write transactions. Accesses are to a 64-bit wide dynamic memory chip-select.

Write buffer enabled

With the write buffer enabled the memory writes 64 bits of data on every clock cycle.

The memory controller re-arbitrates to a different AHB port when data is being written to the buffer. The latency of a buffer transfer might therefore be longer than a nonbuffered transfer.

The worst case additional latency for a buffered transfer is when a buffered INCR16 32-bit wide burst is being performed and the other AHB ports are performing INCR16 64-bit wide transfers. See Table 2-11 for an example of the worse case scenario.

Table 2-11 Write buffer enabled

Clock cycle	AHB 0	AHB 1	AHB 2
1	32-bit write data 0 written into buffer.	AHB port waiting for data	AHB port waiting for data
2	32-bit write data 1 written into buffer. 64-bit write to memory.	AHB port waiting for data	AHB port waiting for data
3	32-bit write data 2 written into buffer.	64-bit write 0 to memory	AHB port waiting for data
4	32-bit write data 3 written into buffer.	64-bit write 1 to memory	AHB port waiting for data
5-17	AHB port waiting for data.	64-bit write 2-14 to memory	AHB port waiting for data
18	AHB port waiting for data.	64-bit write 15 to memory	AHB port waiting for data

Table 2-11 Write buffer enabled (continued)

Clock cycle	AHB 0	AHB 1	AHB 2
19	64-bit write to memory.	Next INCR16 transfer	AHB port waiting for data
20	32-bit write data 4 written into buffer.	AHB port waiting for data	AHB port waiting for data
21	32-bit write data 5 written into buffer.	AHB port waiting for data	64-bit write 0 to memory
23-37	AHB port waiting for data.	AHB port waiting for data	64-bit write 1-14 to memory
38	AHB port waiting for data.	AHB port waiting for data	64-bit write 15 to memory
39	AHB port waiting for data.	AHB port waiting for data	Next INCR16 transfer
40	64-bit write to memory.	AHB port waiting for data	AHB port waiting for data

Note

Table 2-11 on page 2-29 shows the first 39 cycles of the transaction. Cycles 40 to 122 are a continuation of the sequence shown.

Write buffer disabled

Table 2-12 shows that if write buffers are not enabled an INCR16 32-bit wide write burst takes 16 cycles to complete when the write data starts being written to memory.

Table 2-12 Write buffer disabled

Clock cycle	AHB 0	AHB 1	AHB 2
1	32-bit write 0 to memory	AHB port waiting for data	AHB port waiting for data
2-15	32-bit write 2-15 to memory	AHB port waiting for data	AHB port waiting for data
16	32-bit write 16 to memory	AHB port waiting for data	AHB port waiting for data
17	AHB port available	64-bit write 0 to memory	AHB port waiting for data
18-31	AHB port available	64-bit write 1 to memory	AHB port waiting for data
32	AHB port available	64-bit write 2-14 to memory	AHB port waiting for data
33	AHB port available	64-bit write 16 to memory	64-bit write 0 to memory

Table 2-12 Write buffer disabled (continued)

Clock cycle	AHB 0	AHB 1	AHB 2
34-47	AHB port available	AHB port waiting for data	64-bit write 1 to memory
48	AHB port available	AHB port waiting for data	64-bit write 2-14 to memory
49	AHB port available	AHB port available	64-bit write 16 to memory

Zero wait state write transfers

When the buffers are disabled, all write transfers receive **HREADY** LOW wait states until the transfer has started to be performed by the MPMC. **HREADY** then goes HIGH. This functionality is important for multi-port masters to ensure data coherency over multiple ports, or when data is being passed between masters. For example, the ARM11 read and write data ports require the write transfers to receive wait states until the write is being performed to ensure coherency between the two ports. This is the default operation of the MPMC after reset.

With the buffers enabled and empty, a write transfer receives no wait states and completes immediately, enabling reduced write latency if there are no data coherency issues. Subsequent writes are waited until the first write has completed.

To enable zero wait state writes to be performed without using the data merge buffers, the buffers must be enabled and the **HPROT[3:2]** lines driven correctly to indicate that reads are not cacheable and writes are not bufferable.

——— **Note** ———

Exclusive write transfers always have a minimum of two wait states inserted, even when the merge buffer is used.

When the buffers are disabled, write bursts have wait states added after the first nonsequential transfer until the MPMC starts performing the transfer, as if it were a single transfer. The sequential transfers might then have wait states inserted, depending on the width and length of the burst, and whether data merging is enabled.

2.1.6 Endian and packing logic

The endian and packing block performs the following:

- little-endian and big-endian conversion for ARMv5
- AHB byte strobe handling for ARM11 unaligned data support
- data packing within the read and write merge buffers.

2.1.7 Arbiter

The arbiter arbitrates between the AHB slave memory interfaces. AHB port 0 has the highest access priority, and the MBX Interface Port has the lowest priority. For more information see *Arbitration* on page 2-43.

2.1.8 Memory controller state machine

The memory controller state machine comprises two functional blocks:

- a static memory controller (including the NAND flash controller)
- a dynamic memory controller.

Low transaction latency and high memory bandwidth are conflicting design parameters.

A memory controller designed to support high bandwidth has logic to reorder transactions to maximize memory efficiency. This logic increases transaction latency.

A memory controller designed to reduce latency has less logic for the transaction to pass through, reducing the amount of logic used to maximize the transaction efficiency. This decreases the supported memory bandwidth.

The memory controllers have been designed with both these parameters in mind, and have both good latency and memory bandwidth.

Providing multiple AHB memory ports in the memory controller enables the memory bandwidth to be shared over multiple AHB buses. This reduces the load on performance-critical buses.

To make full use of dynamic memory bandwidth a multiple-port design is required so that:

- the transaction order can be rearranged to maximize the number of in page accesses
- the dynamic memory transactions can be pipelined
- the dynamic memory transactions can be interleaved.

2.1.9 Pad interface

The pad interface block provides the interface to the pads. The pad interface uses a feedback clock, **MPMCFBCLKIN[7:0]**, to capture read data for SDR-SDRAM memory devices. To capture read data for DDR-SDRAM devices, the data strobe signals, **MPMCDQSIN[3:0]** and **nMPMCDQSIN[3:0]**, are used together with an external *Delay Locked Loop* (DLL) block to resynchronize from the off-chip to on-chip domains.

Figure 2-13 on page 2-33 shows a block diagram of the pad interface.

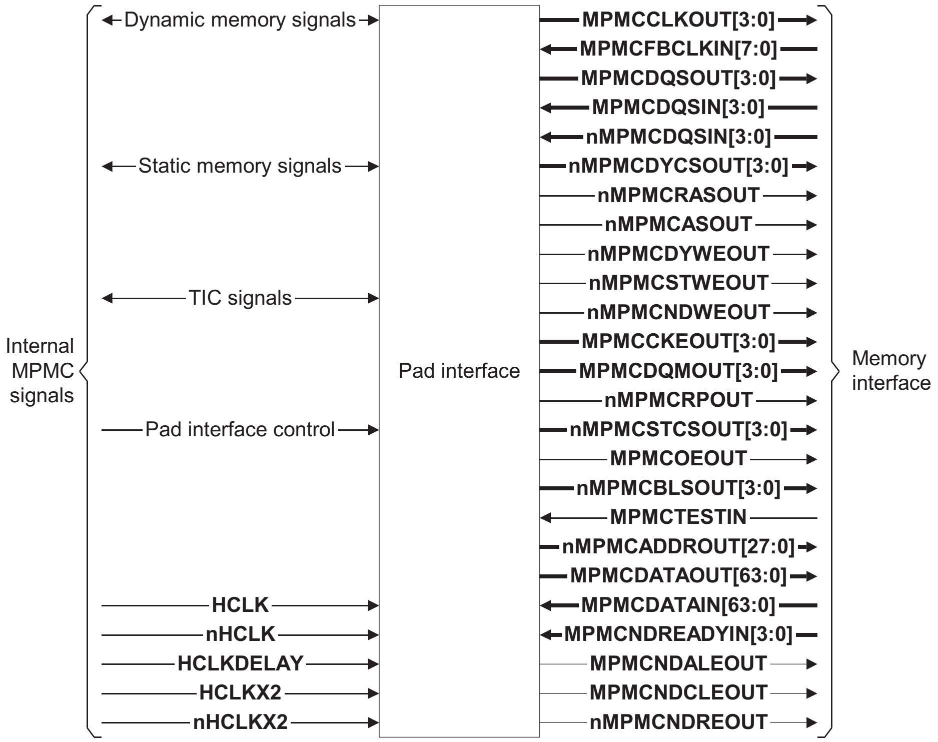


Figure 2-13 Pad interface block diagram

2.1.10 Test Interface Controller (TIC)

The TIC enables TIC device testing. For full details about the TIC, see the *AMBA Specification*. The AHB master interface must normally be placed on the same AHB interface as the ARM processor. Figure 2-14 on page 2-34 shows a block diagram of the TIC.

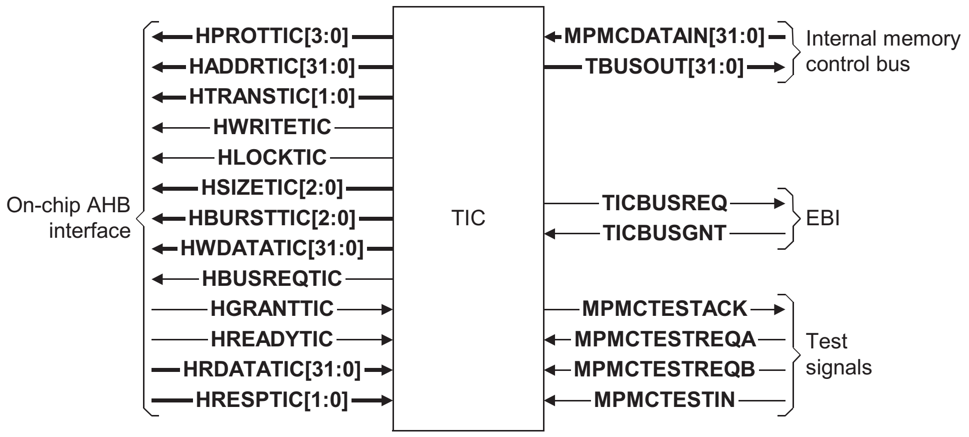


Figure 2-14 TIC block diagram

2.2 Overview of a MPMC, ASIC, or ASSP system

Figure 2-15 shows the MPMC (GX176) in an example system.

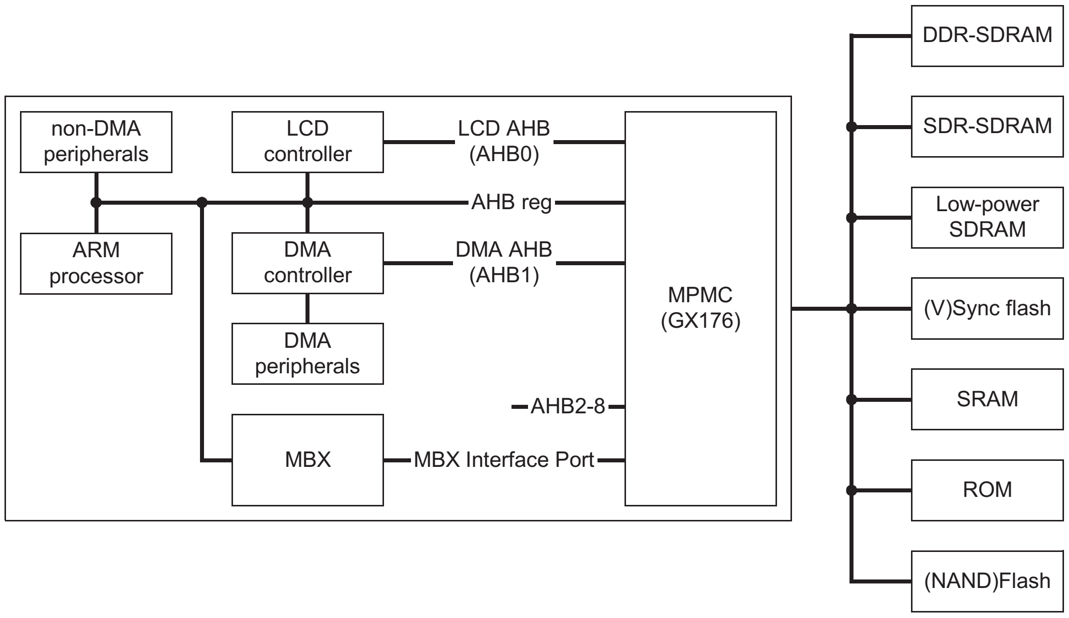


Figure 2-15 MPMC (GX176) in an example system

The example system uses two types of buses as described in:

- *External bus*
- *Internal bus* on page 2-36.

2.2.1 External bus

The off-chip bus that contains data, address, and control signals, connects the ASIC or ASSP to the external memory.

Note

- Connecting a large number of memory devices externally impacts on performance because of signal loading.
- Only one memory device can be accessed at a time.

2.2.2 Internal bus

The on-chip bus enables communication between the on-chip peripherals. The MPMC appears as a standard slave on the on-chip bus and controls the memory on the external bus.

Providing multiple AHB interfaces improves system performance by enabling several access requests to be presented to the memory controller at the same time. This enables the MPMC to pipeline many of the operations (for example, bank activate and precharge), and so reduce the average system access latency and improve utilization of external memory. The use of multiple AHB interfaces also improves system performance by removing heavy DMA traffic from the main AHB bus.

2.3 AHB slave memory interface priority

AHB interface 0 has the highest access priority, followed by AHB interface 1 through to AHB interface 8. The MBX interface has the lowest priority.

2.3.1 AHB memory port latency

Each AHB memory port has a programmable TimeOut register. When a memory request is made to the port the value of the counter is loaded. Every cycle where the port transaction is not serviced the TimeOut register counts down. When the TimeOut counter reaches zero, the priority of the port is increased to a level higher than all ports which have not timed out. This functionality enables each AHB memory port be programmed with a deterministic latency. This also enables the amount of bandwidth that a port consumes to be indirectly defined.

2.4 Low power operation

In many systems, the contents of the memory system have to be maintained during low-power sleep modes. The MPMC provides two features to enable this:

- dynamic memory refresh over soft reset
- a mechanism to place the dynamic memories into self-refresh mode.

Self-refresh mode can be entered automatically by hardware or manually by software:

- It can be entered manually setting the SREFREQ bit in the MPMCDynamicControl register and polling the SREFACK bit in the MPMCStatus register.
- It can be entered automatically using a *Power Management Unit* (PMU). This is typically present to control the safe transition between the following modes:
  - power-up
  - reset
  - normal
  - sleep.

The PMU can be used to enable self-refresh mode to be entered automatically. To do this, the PMU asserts the **MPMCSREFREQ** signal when self-refresh mode is to be entered. The memory controller then closes any open memory banks and puts the external memory into self-refresh mode. The MPMC then asserts the **MPMCSREFACK** signal to indicate to the PMU that self-refresh mode is entered. The system must ensure that the memory subsystem is idle before asserting **MPMCSREFREQ**. The memory controller accepts transactions to dynamic memory even in self-refresh mode, if the accesses are targeted to chip selects that are populated with SyncFlash and its variants. This is required to aid booting from a SyncFlash device. Deasserting **MPMCSREFREQ** returns the memory to normal operation. See the memory data sheet for refresh requirements.

Note

- If **MPMCSREFREQ** is not required this signal must be tied LOW.
- Static memory can be accessed as normal when the SDRAM memory is in self-refresh mode.

2.4.1 Low-power SDRAM deep sleep mode

The MPMC supports JEDEC low-power SDRAM deep-sleep mode. Deep-sleep mode can be entered by setting the deep-sleep mode (DP) bit in the MPMCDynamicControl register. The device is then put into a low-power mode where the device is powered down and no longer refreshed. All data in the memory is lost.

2.4.2 Low-power SDRAM partial array refresh

The MPMC supports JEDEC low-power SDRAM partial array refresh. Partial array refresh can be programmed by initializing the SDRAM memory device appropriately. When the memory device is put into self-refresh mode only the memory banks specified are refreshed. The memory banks that are not refreshed lose their data contents.

2.5 Lock and semaphores

Locked accesses on the AHB bus, transactions where **HMASTLOCK** is HIGH, are processed appropriately. When the memory controller performs a locked transfer the memory controller does not rearbitrate to another AHB port until the lock transfer is completed.

2.6 Burst types

All AHB burst types are supported.

——— **Note** ———

INCR transfers are split internally into four-word bursts.

————————

2.7 Busy transfer type

When an AHB master generates busy (AHB **HTRANS**=BUSY) cycles the memory controller waits until busy is inactive before completing the transfer. This increases transfer latency. The memory controller does not rearbitrate to another AHB port if busy goes active.

2.8 Arbitration

The MPMC is normally used with multi-level AHB and no arbitration is required. This section describes the MPMC AHB memory port arbitration strategy when it is used.

Note

- It is recommended that the AHB arbiter, where present, rearbitrates on a burst boundary because this leads to the most efficient bus utilization.
- AHB protocol does not enable AHB masters to break a defined length burst transfer (INCR4, WRAP4, INCR8, WRAP8, INCR16, and WRAP16). An AHB master can only break an undefined length burst transfer (INCR).
- It is not possible for an AHB burst to cross an SDRAM column, because AHB bursts must not cross a 1KB boundary, and the smallest SDRAM column length supported is 1KB long.

2.8.1 Occurrence

The arbitration occurrence is the time when the memory controller arbitrates. The following lists AHB transfers that affect arbitration:

- If an 8, 16, or 32-bit wide AHB burst is submitted and the AHB port buffers are enabled, the memory controller can arbitrate to a different AHB port when data is being written to or read from the buffer.

  Note

  The longest AHB transfer is 16 AHB transfers long (INCR16/WRAP16).

- When a locked transfer (as defined by the AHB **HLOCK** signal) has started the AHB memory port is not arbitrated until the locked transfer has completed and the AHB lock signal is deasserted.
- When an AHB burst transfer has started the AHB memory port is not arbitrated until the burst has completed.
- The memory controller does not arbitrate to another AHB port if **HTRANS** = BUSY.

2.8.2 Priority

The following lists the arbitration scheme:

- The highest priority AHB memory port is a port where the TimeOut register has timed out.If more than one ports has timed out, the port with the lowest port number is selected. For example, port 0 is selected over port 1.
- If none of the AHB memory ports have timed out then the next highest priority port is for the port where the transaction is to an already open SDRAM memory row.

  If the transaction of more than one port is to an already open SDRAM memory row then the port with the lowest port number is selected.
- For the case where none of the AHB memory ports have timed out and none of the accesses are too open SDRAM memory rows. The memory request to the lowest port number is selected. In other words port 0 is selected over port 1.

  Note

  When a buffered 8, 16 or 32-bit wide transfer is in progress the memory controller does not rearbitrate to a different AHB port if the access is to an unopened SDRAM row.

2.8.3 AHB memory port latency

Each AHB memory port has a programmable TimeOut register. When a memory request is made to the port the value of the counter is loaded. Every cycle where the transaction of the port is not serviced the TimeOut register counts down. When the Time Out counter reaches zero, the priority of the port is increased. This functionality enables each AHB memory port be programmed with a deterministic latency. This also enables the amount of bandwidth that a port consumes to be indirectly defined.

2.8.4 Arbitration strategy

The bandwidth and latency seen by masters on the AHB data ports varies depending on the port they are connected to, and whether timeouts are enabled for that port:

- devices which regularly use large amounts of bandwidth, such as video encoders/decoders, must be connected to the highest priority ports
- devices which can require high bandwidth, but randomly, such as a CPU, must be connected to medium priority ports

- devices which require low bandwidth or which do not have to access memory very often, such as DMA controllers, must be connected to the lower priority ports.

If a master requires a fixed bandwidth, such as an audio or video device which does not function correctly with a lower bandwidth, timeouts for the ports must be programmed appropriately to ensure that they are granted often enough to satisfy the bandwidth requirement.

2.9 Worst case transaction latency

The worst case transaction latency for the highest priority AHB memory port is 58 clock cycles. The worst case latency of the lower priority AHB memory ports is TimeOut + 58 cycles, assuming that another higher priority port has not timed out. These values are based on the assumptions given in the following sections:

- *Worst case transaction latency for the highest priority AHB memory port*
- *Worst case transaction latency for the lower priority AHB memory ports* on page 2-47
- *System factors affecting worst case latency* on page 2-49.

2.9.1 Worst case transaction latency for the highest priority AHB memory port

The following assumptions were made for calculating the worst case transaction latency:

- System factors are ignored in these calculations. For information on system factors see *System factors affecting worst case latency* on page 2-49.
- SDRAM memory latency values are:
  - precharge = 3
  - active = 3
  - CAS = 3
  - auto-refresh (t_{RFC}) = 7.
- The SDRAM memory chip selects have a 64-bit wide databus.
- There are no devices connected to the static memory chip selects.
- AHB port 0 TimeOut register is programmed to be less than or equal to the slowest transfer. This is normally either a INCR16 or WRAP16 read to an unopened SDRAM page.

Note

Using SDRAM memory chip selects with a 16-bit or 32-bit wide databus, SDRAM memory with larger latency values, or using slow static memory devices in a system might impact the worst case latency.

For the MPMC the worst case latency scenario is as follows:

- A lower priority port is performing an INCR16 read request. The read data is to a different row from the one already opened. Therefore a precharge, activate, and read command must be sent to the SDRAM.

- An auto-refresh is generated after the INCR16 read is submitted.
- The highest priority AHB memory port performs an INCR16 read transaction. The read data is to a different row from the one already opened, therefore a precharge, activate, and read command must be sent to the SDRAM.

The read data is to unopened SDRAM memory rows.

Before the first data from the highest priority AHB memory port is returned the following transactions occur:

1. The INCR16 read is performed. The read data is to a different row from the one already opened. This transaction takes 30 cycles.
2. The SDRAM auto-refresh is generated. This transaction takes 13 cycles.
3. The AHB memory port 0 priority read transaction is performed. The read data is to a different row from the one already opened. The first data is returned after 15 cycles. The following 15 transfers of the burst take an additional 15 cycles to be performed.

Worst case transaction latency for highest priority AHB memory port = 30+13+15 = 58 cycles (first data returned).

Worst case latency for highest priority AHB memory port to complete = 30+13+15+15 = 73 cycles (last piece of longest AHB data returned).

2.9.2 Worst case transaction latency for the lower priority AHB memory ports

The following assumptions were made for calculating the worst case transaction latency:

- System factors are ignored in these calculations. For information on system factors see *System factors affecting worst case latency* on page 2-49.
- SDRAM memory latency values are:
  - — precharge = 3
  - — active = 3
  - — CAS = 3
  - — auto-refresh (t_{RFC}) = 7.
- The SDRAM memory chip selects have a 64-bit wide databus.
- There are no devices connected to the static memory chip selects.

- The required AHB port latency is programmed into the appropriate TimeOut register. The value programmed is normally the required timeout latency minus the latency of the slowest burst transfer.
- No other AHB ports have their TimeOut registers programmed.

Note

Using SDRAM memory chip selects with a 16-bit or 32-bit wide databus, SDRAM memory with larger latency values, or using slow static memory devices in a system might impact the worst case latency.

For the MPMC the worst case latency for the lower priority memory port scenario is as follows:

- The AHB memory port that you are interested in submits a read transaction. However this transaction is not performed because there are higher priority transactions. The read data is to a different row from the one already opened. Therefore a precharge, activate, and read command must be sent to the SDRAM.
- An INCR16 read is submitted to a higher priority port. The read data is to unopened SDRAM memory rows.
- The TimeOut register for the AHB memory port you are interested in times out.
- An auto-refresh is generated after the INCR16 read is submitted.
- The AHB memory port you are interested in performs a read access. The read data is to a different row from the one already opened. Therefore a precharge, activate, and read command must be sent to the SDRAM.

The following transactions occur before the first data is returned from the port you are interested in:

1. Read transaction submitted.
2. The INCR16 read is performed for higher priority port. The read data is to a different row from the one already opened. This transaction takes 30 cycles.
3. The TimeOut counter for the port you are interested in counts to 0.
4. The SDRAM auto-refresh is generated. This transaction takes 13 cycles.
5. The highest priority read transaction is performed. The read is to an unopened SDRAM memory row. The first data is returned after 15 cycles. The following 15 transfers of the burst take an additional 15 cycles to be performed.

Worst case transaction latency for highest priority AHB memory port = 30+13+15 = 58 cycles.

Worst case latency for highest priority AHB memory port to complete = 30+13+15+15 = 73 cycles.

Note

The latency for the MBX Interface port is similar to that for the lower AHB ports but there is no Timeout Register available.

2.9.3 System factors affecting worst case latency

This section describes the system factors that can affect the memory controller worst case latency to a memory request.

MPMC AHB Memory port priority

The memory controller AHB memory ports are prioritized so that the most efficient transfers are performed first. High priority ports, AHB ports with a low value, are selected in preference to lower priority ports. The AHB TimeOut register enables the maximum latency for a port to be programmed.

AHB bus

If there are multiple AHB masters on the same AHB bus, then a master can only receive ownership of the bus, and submit transactions, when the AHB arbiter grants the bus to the master. The arbitration strategy in the AHB arbiter then affects the worst case latency for a system with multiple masters on an AHB bus.

AHB masters

When an AHB master generates busy (AHB **HTRANS**=BUSY) cycles the memory controller waits until busy is inactive before completing the transfer. This increases transfer latency. The memory controller does not re-arbitrate to another AHB port if busy goes active.

If an AHB master performs locked (AHB **HMASTLOCK**=LOCK) cycles this means that the memory controller cannot re-arbitrate until the locked transaction has completed. This can then increase worst case transfer latency, because the memory controller is not able to re-arbitrate to a higher priority port until the locked access has completed.

Memory devices

If slow SDRAM memories and/or a 16 or 32-bit SDRAM chip select are used, transactions take longer to complete. This affects transfer latency.

Pad multiplexing

If the memory bus is multiplexed externally, for example, by using an *External Bus Interface* (EBI), the worst case transfer latency is affected because the external bus is shared by multiple devices.

2.10 Sharing memory bandwidth between AHB ports

By programming the AHB port TimeOut values appropriately the bandwidth of the memory controller can be shared between the AHB ports of the memory controller.

Note

The MBX Interface port does not have a Timeout Register so the AHB ports must be configured so that the MBX Interface port meets the bandwidth requirement.

2.10.1 Typical AHB port TimeOut value given bandwidth requirement

To meet a given bandwidth requirement the TimeOut value must be programmed less than the value provided by the following formula:

$$\text{TimeOut value} = \frac{\text{(AHB frequency) x (number of bytes in average AHB burst)}}{\text{(required AHB bandwidth)}} - \text{(number of transactions in AHB burst)}$$

Note

This formula assumes that the AHB masters on each of the AHB ports always have requests pending.

2.10.2 Typical AHB port TimeOut value given percentage of memory bandwidth requirement

To compute the TimeOut values that must be programmed for the case where each AHB port requires a percentage of the total memory bandwidth, first the expected memory bandwidth of the system must be calculated. From this the bandwidth per port can be calculated. Finally you can use the formula in *Typical AHB port TimeOut value given bandwidth requirement*.

2.10.3 Typical AHB bandwidth requirement example

This section provides an example of how to program the AHB port TimeOut registers given multiple AHB ports and their bandwidth requirements.

System

The memory controller, AHB bus, and SDR-SDRAM memory operate at 100MHz. The SDR-SDRAM has a 64-bit wide databus.

Bandwidth requirement

The AHB bandwidth requirements are:

- AHB port 0 requires at least 40MB/s
- AHB port 1 requires at least 20MB/s
- AHB port 2 requires at least 2MB/s.

The AHB burst types are:

- AHB port 0 normally performs AHB INCR16, 64-bit wide transfers
- AHB port 1 normally performs AHB INCR8, 32-bit wide transfers
- AHB port 2 normally performs AHB INCR4, 16-bit wide transfers.

Calculations required

The TimeOut values can be calculated as indicated below.

AHB port 0 TimeOut value Each INCR16 transfer provides 128 bytes of data. Therefore there are 327,680 (40MB divided by 128B) INCR16 bursts required per second to satisfy the minimum bandwidth requirement. On average there are 305 clock cycles (100M clock cycles divided by 327,680) between AHB port 0 transactions to meet the requirement. Each transaction takes about 16 cycles to complete.

$$\text{TimeOut value} = \frac{\text{100MHz x 128}}{\text{40MB}} - 16 = \text{289 cycles}$$

Therefore the AHB TimeOut value for AHB port 0 must be programmed to less than 289 clock cycles.

AHB port 1 TimeOut value Each INCR8 transfer provides 32 bytes of data. Therefore there are 655,360 (20MB divided by 32B) INCR8 bursts required per second to satisfy the minimum bandwidth requirement. On average there are 152 clock cycles (100M clock cycles divided by 655,360) between AHB port 1 transactions to meet the requirement. Each transaction takes about eight cycles to complete.

$$\text{TimeOut value} = \frac{\text{100MHz x 32}}{\text{20MB}} - 8 = \text{144 cycles}$$

Therefore the AHB TimeOut value for AHB port 1 must be programmed to less than 144 clock cycles.

AHB port 2 TimeOut value Each INCR4 transfer provides eight bytes of data. Therefore there are 262,144 (2MB divided by 8B) INCR4 bursts required per second to satisfy the minimum bandwidth requirement. On average there are 381 clock cycles (100M clock cycles divided by 262,144) between AHB port two transactions to meet the requirement. Each transaction takes about four cycles to complete.

$$\text{TimeOut value} = \frac{100\text{MHz} \times 8}{2\text{MB}} - 4 = 377 \text{ cycles}$$

Therefore the AHB TimeOut value for AHB port 1 must be programmed to less than 377 clock cycles.

2.10.4 Typical AHB percentage bandwidth example

This section provides an example of how to program the AHB port TimeOut registers given multiple AHB ports and their percentage of bandwidth requirements.

System

The memory controller, AHB bus, and SDR-SDRAM memory operate at 100MHz. The SDR-SDRAM has a 64-bit wide databus.

Bandwidth requirement

The AHB bandwidth requirements are:

- AHB port 0 requires 10% of memory bandwidth minimum
- AHB port 1 requires 5% of memory bandwidth minimum
- AHB port 2 requires 0.5% of memory bandwidth minimum.

The AHB burst types are:

- AHB port 0 normally performs AHB INCR16, 64-bit wide transfers
- AHB port 1 normally performs AHB INCR8, 32-bit wide transfers
- AHB port 2 normally performs AHB INCR4, 16-bit wide transfers.

Calculations required

The TimeOut values can be calculated as indicated below.

Compute the real system bandwidth Theoretically the SDRAM memory can provide 781MB/s. However, we assume that the real system bandwidth is 400MB/s because of the system burst types and page hit rates.

Note

The theoretical value of 781MB/s is obtained by multiplying 100MHz by 8 bytes (64-bit), and dividing by 1024 twice to convert into MB.

Calculate the memory bandwidth per AHB port For AHB port 0 10% of the 400MB/s memory controller bandwidth is 40MB/s.

For AHB port 1 5% of the 400MB/s memory controller bandwidth is 20MB/s.

For AHB port 2 0.5% of the 400MB/s memory controller bandwidth is 2MB/s.

Compute the TimeOut value using the following formula

$$\text{TimeOut value} = \frac{\text{(AHB frequency) x (number of bytes in average AHB burst)}}{\text{(required AHB bandwidth)}} - \text{(number of transactions in AHB burst)}$$

For the AHB port 0 the TimeOut value is:

$$\text{TimeOut value} = \frac{\text{100MHz x 128}}{\text{40MB}} - 16 = 289 \text{ cycles}$$

For the AHB port 1 the TimeOut value is:

$$\text{TimeOut value} = \frac{\text{100MHz x 32}}{\text{20MB}} - 8 = 144 \text{ cycles}$$

For the AHB port 2 the TimeOut value is:

$$\text{TimeOut value} = \frac{\text{100MHz x 8}}{\text{2MB}} - 4 = 377 \text{ cycles}$$

2.11 Memory bank select

Eight independently-configurable memory chip selects are supported, with a separate AMBA AHB select, **HSELMPMCxCS[7:0],** to select the appropriate chip select:

- **HSELMPMCxCS** selects 0-3 are used to select static memory devices
- **HSELMPMCxCS** selects 4-7 are used to select dynamic memory devices.

The MBX Interface port only requires access to dynamic memory devices using input signals **GSELCS[7:4]**.

Table 2-13 shows the relationship between the AHB select, **HSELMPMCxCS[7:0]**, and the memory chip selects. In the table an x refers to the AHB port number.

Table 2-13 Memory bank selection

AHB select	Chip select	Memory device	Memory chip select
HSELMPMCxCS[0]	0	Static Mem 0	**nMPMCSTCSOUT[0]**
HSELMPMCxCS[1]	1	Static Mem 1	**nMPMCSTCSOUT[1]**
HSELMPMCxCS[2]	2	Static Mem 2	**nMPMCSTCSOUT[2]**
HSELMPMCxCS[3]	3	Static Mem 3	**nMPMCSTCSOUT[3]**
HSELMPMCxCS[4]	4	Dynamic Mem 0	**nMPMCDYCSOUT[0]**
HSELMPMCxCS[5]	5	Dynamic Mem 1	**nMPMCDYCSOUT[1]**
HSELMPMCxCS[6]	6	Dynamic Mem 2	**nMPMCDYCSOUT[2]**
HSELMPMCxCS[7]	7	Dynamic Mem 3	**nMPMCDYCSOUT[3]**

The amount of memory allocated to a chip select is determined by the system AHB decoder. A further select line, **HSELMPMCxG**, is provided at each AHB slave port to select the MPMC slave. The largest amount of memory that can be allocated to a single chip select is 256Mb.

——— **Note** ———

Chip selects connected to Micron SyncFlash or V-SyncFlash must have **HADDR[28:27]** available for use as the software control for generating *Hardware Command Sequences* (HCS). If HCSs are used, the AHB decoder can only use **HADDR[31:29]** for decoding the chip select. If HCSs are not used, the chip select must be located at an address where **HADDR[28]** is zero.

2.12 Memory map

The MPMC provides hardware support for booting from external nonvolatile memory. During booting the nonvolatile memory must be located at address `0x00000000` in memory. When the system is booted the SRAM or SDRAM memory can be remapped to address `0x00000000` by modifying the address map in the AHB decoder.

2.12.1 Chip select 1 static memory configuration

The memory width, chip select polarity, and byte lane select polarity of static memory chip select 1 can be configured by using a number of input tie-off signals. This enables you to boot from chip select 1.

The configuration tie-off signals are:

- **MPMCSTCS1MW[1:0]**, memory width select
- **MPMCSTCS1POL**, chip select polarity
- **MPMCSTCS1PB**, byte lane select polarity.

2.12.2 Chip select 5 SDRAM memory configuration

The address mapping and memory device type of SDRAM memory chip select 5 can be configured by using a number of input tie-off signals. This enables you to boot from chip select 5.

The configuration tie-off signals are:

- **MPMCDYCS5ADRMAP[7:0]**, address mapping
- **MPMCDYCS5DEVICE[2:0]**, memory device
- **MPMCDYCS5CASDLY[3:0]**, CAS delay for chip select 5
- **MPMCDYSDRPOL**, polarity of the flip-flop in which the read data is first captured
- **MPMCDYSDRDLY[1:0]**, SDR clocking scheme for data capture.

2.12.3 Boot from flash, SRAM remapped after boot

The system set up is:

- chip select 1 is connected to the boot flash device
- chip select 0 is connected to the SRAM to be remapped to `0x00000000` after boot
- the AHB decoder contains the functionality to remap the address.

The boot sequence is as follows:

1. At power on the reset chip select 1 is located at `0x00000000` in the AHB address map. The following signals are configured so that the nonvolatile memory device can be accessed:
   - **MPMCSTCS1MW[1:0]**
   - **MPMCSTCS1PB**
   - **MPMCSTCS1POL** (also **MPMCSTCS0POL**, **MPMCSTCS2POL**, and **MPMCSTCS3POL**).
2. When the power-on reset (**nPOR**) and AHB reset (**HRESETn**) go inactive, the processor starts booting from `0x00000000` in memory.
3. The software programs the optimum delay values in the flash memory so that the boot code can run at full speed.
4. The other chip selects are initialized.
5. The AHB decoder address map is modified so that the SRAM is located at address `0x00000000`.
6. The ARM reset and interrupt vectors are copied from flash memory to SRAM that can then be accessed at address `0x00000000`.
7. More boot, initialization, or application code is executed.

2.12.4 Example of a boot from flash, SDRAM remapped after boot

The system set up is:

- chip select 1 is connected to the boot flash device
- chip select 4 is connected to the SDRAM to be remapped to `0x00000000` after boot
- the AHB decoder contains the functionality to remap the address.

The boot sequence is as follows:

1. At power on the reset chip select 1 is located at `0x00000000` in the AHB address map. The following signals are configured so that the nonvolatile memory device can be accessed:
   - **MPMCSTCS1MW[1:0]**
   - **MPMCSTCS1PB**
   - **MPMCSTCS1POL** (also **MPMCSTCS0POL**, **MPMCSTCS2POL**, and **MPMCSTCS3POL**).

2. When the power-on reset (**nPOR**) and AHB reset (**HRESETn**) go inactive, the processor starts booting from `0x00000000` in memory.
3. The software programs the optimum delay values in the flash memory so that the boot code can run at full speed.
4. The other chip selects are initialized.
5. The AHB decoder address map is modified so that the SDRAM is located at address `0x00000000`.
6. The ARM reset and interrupt vectors are copied from flash memory to SDRAM that can then be accessed at address `0x00000000`.
7. More boot, initialization, or application code is executed.

2.12.5 Memory aliasing

Memory aliasing is allowed.

2.12.6 Unused AHB HADDRx address bits

If some of the **HADDRx** address bits are not required, the unused address bits must be tied LOW.

2.12.7 External memory turnaround

To prevent bus contention on the external memory data bus, a turnaround time is required between static and dynamic memory accesses. The minimum times are:

- static to dynamic = 1 cycle
- dynamic to static = 3 cycles.

The MPMCStaticWaitTurn0-3 Registers, which control the turnaround times between static memory accesses, can also be used to extend the static to dynamic turnaround time.

Note

The dynamic to static turnaround time might be longer than the minimum if the dynamic controller has to perform SDRAM commands after the data access.

The dynamic memory controller has higher priority over the external data bus than the static memory controller, so it is possible that a burst of transfers to static memory can be broken by dynamic memory transfers. If this occurs, there will be a turnaround time on either side of all dynamic accesses.

2.13 Sharing memory interface signals

The memory interface signals, and the memory controller primary input and output signals, can be shared with other peripherals by using an EBI block. For more information on the EBI see the *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual*.

Chapter 3
Programmer's Model

This chapter describes the MPMC (GX176) registers and provides details required when programming the device. It contains the following sections:

3.1 About the programmer's model

The base address of the MPMC is not fixed, but is determined by the AHB decoder, and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed. The registers of the MPMC can only be accessed using the AHB register interface port of the memory controller.

The external memory is accessed using the AHB memory interface ports. Addresses are not fixed, but are determined by the AHB decoder, and can be different for any particular system implementation. Transfers to the external memories of the MPMC are selected by the **HSELMPMC[8:0]CS[7:0]** signals for the AHB ports, and the **GSELCS[7:4]** signals for the MBX interface port.

Note

[8:0] indicates the AHB port number, and [7:0] indicates the chip select to be accessed.

Note

AHB masters that are 64 bits wide cannot use load or store multiple instructions to access the control registers, and writes must be aligned to the transfer size. For ARM11, the control registers must be placed in device memory space (ARMv6 memory region type).

3.2 Register summary

Table 3-1 summarizes the MPMC (GX176) registers.

Table 3-1 MPMC register summary

Name	Offset from base	Type	Reset HRESETn	Reset nPOR	Description
MPMCControl	`0x000`	RW	`0x1`	-	See *Control Register* on page 3-10
MPMCStatus	`0x004`	RO	-	`0x15`	See *Status Register* on page 3-11
MPMCConfig	`0x008`	RW	-	`0x0`	See *Configuration Register* on page 3-12
MPMCDynamicControl	`0x020`	RW	-	`0x000E`	See *Dynamic Memory Control Register* on page 3-13
MPMCDynamicRefresh	`0x024`	RW	-	`0x000`	See *Dynamic Memory Refresh Timer Register* on page 3-16
MPMCDynamicReadConfig	`0x028`	RW	-	`0x----`[a]	See *Dynamic Memory Read Configuration Register* on page 3-17
MPMCDynamictRP	`0x030`	RW	-	`0xF`	See *Dynamic Memory Precharge Command Period Register* on page 3-18
MPMCDynamictRAS	`0x034`	RW	-	`0xF`	See *Dynamic Memory Active To Precharge Command Period Register* on page 3-19
MPMCDynamictSREX	`0x038`	RW	-	`0x7F`	See *Dynamic Memory Self-refresh Exit Time Register* on page 3-20
MPMCDynamictWR	`0x044`	RW	-	`0xF`	See *Dynamic Memory Write Recovery Time Register* on page 3-21
MPMCDynamictRC	`0x048`	RW	-	`0x1F`	See *Dynamic Memory Active To Active Command Period Register* on page 3-22
MPMCDynamictRFC	`0x04C`	RW	-	`0x1F`	See *Dynamic Memory Auto-refresh Period Register* on page 3-22
MPMCDynamictXSR	`0x050`	RW	-	`0xFF`	See *Dynamic Memory Exit Self-refresh Register* on page 3-23
MPMCDynamictRRD	`0x054`	RW	-	`0xF`	See *Dynamic Memory Active Bank A-B Time Register* on page 3-24

Table 3-1 MPMC register summary (continued)

Name	Offset from base	Type	Reset HRESETn	Reset nPOR	Description
MPMCDynamictMRD	`0x058`	RW	-	`0xF`	See *Dynamic Memory Load Mode Register* on page 3-25
MPMCDynamictCDLR	`0x05C`	RW	-	`0xF`	See *Dynamic Memory Last Data In To Read Command Time Register* on page 3-26
MPMCStaticExtendedWait	`0x080`	RW	-	`0x000`	See *Static Memory Extended Wait Register* on page 3-26
MPMCDynamicConfig0	`0x100`	RW	-	`0x000000`	See *Dynamic Memory Configuration Registers 0-3* on page 3-27
MPMCDynamicRasCas0	`0x104`	RW	-	`0x783`	See *Dynamic Memory RAS and CAS Delay Registers 0-3* on page 3-32
MPMCDynamicConfig1	`0x120`	RW	-	`0x00----`[a]	See *Dynamic Memory Configuration Registers 0-3* on page 3-27
MPMCDynamicRasCas1	`0x124`	RW	-	`0x--3`[a]	See *Dynamic Memory RAS and CAS Delay Registers 0-3* on page 3-32
MPMCDynamicConfig2	`0x140`	RW	-	`0x000000`	See *Dynamic Memory Configuration Registers 0-3* on page 3-27
MPMCDynamicRasCas2	`0x144`	RW	-	`0x783`	See *Dynamic Memory RAS and CAS Delay Registers 0-3* on page 3-32
MPMCDynamicConfig3	`0x160`	RW	-	`0x000000`	See *Dynamic Memory Configuration Registers 0-3* on page 3-27
MPMCDynamicRasCas3	`0x164`	RW	-	`0x783`	See *Dynamic Memory RAS and CAS Delay Registers 0-3* on page 3-32
MPMCStaticConfig0	`0x200`	RW	-	`0x0000-0`[a]	See *Static Memory Configuration Registers 0-3* on page 3-33
MPMCStaticWaitWen0	`0x204`	RW	-	`0x0`	See *Static Memory Write Enable Delay Registers 0-3* on page 3-35
MPMCStaticWaitOen0	`0x208`	RW	-	`0x0`	See *Static Memory Output Enable Delay Registers 0-3* on page 3-36
MPMCStaticWaitRd0	`0x20C`	RW	-	`0x1F`	See *Static Memory Read Delay Registers 0-3* on page 3-37

Table 3-1 MPMC register summary (continued)

Name	Offset from base	Type	Reset HRESETn	Reset nPOR	Description
MPMCStaticWaitPage0	`0x210`	RW	-	`0x1F`	See *Static Memory Page Mode Read Delay Registers 0-3* on page 3-37
MPMCStaticWaitWr0	`0x214`	RW	-	`0x1F`	See *Static Memory Write Delay Registers 0-3* on page 3-38
MPMCStaticWaitTurn0	`0x218`	RW	-	`0xF`	See *Static Memory Turnaround Delay Registers 0-3* on page 3-39
MPMCStaticConfig1	`0x220`	RW	-	`0x0000--`[a]	See *Static Memory Configuration Registers 0-3* on page 3-33
MPMCStaticWaitWen1	`0x224`	RW	-	`0x0`	See *Static Memory Write Enable Delay Registers 0-3* on page 3-35
MPMCStaticWaitOen1	`0x228`	RW	-	`0x0`	See *Static Memory Output Enable Delay Registers 0-3* on page 3-36
MPMCStaticWaitRd1	`0x22C`	RW	-	`0x1F`	See *Static Memory Read Delay Registers 0-3* on page 3-37
MPMCStaticWaitPage1	`0x230`	RW	-	`0x1F`	See *Static Memory Page Mode Read Delay Registers 0-3* on page 3-37
MPMCStaticWaitWr1	`0x234`	RW	-	`0x1F`	See *Static Memory Write Delay Registers 0-3* on page 3-38
MPMCStaticWaitTurn1	`0x238`	RW	-	`0xF`	See *Static Memory Turnaround Delay Registers 0-3* on page 3-39
MPMCStaticConfig2	`0x240`	RW	-	`0x0000-0`[a]	See *Static Memory Configuration Registers 0-3* on page 3-33
MPMCStaticWaitWen2	`0x244`	RW	-	`0x0`	See *Static Memory Write Enable Delay Registers 0-3* on page 3-35
MPMCStaticWaitOen2	`0x248`	RW	-	`0x0`	See *Static Memory Output Enable Delay Registers 0-3* on page 3-36
MPMCStaticWaitRd2	`0x24C`	RW	-	`0x1F`	See *Static Memory Read Delay Registers 0-3* on page 3-37
MPMCStaticWaitPage2	`0x250`	RW	-	`0x1F`	See *Static Memory Page Mode Read Delay Registers 0-3* on page 3-37

Table 3-1 MPMC register summary (continued)

Name	Offset from base	Type	Reset HRESETn	Reset nPOR	Description
MPMCStaticWaitWr2	`0x254`	RW	-	`0x1F`	See *Static Memory Write Delay Registers 0-3* on page 3-38
MPMCStaticWaitTurn2	`0x258`	RW	-	`0xF`	See *Static Memory Turnaround Delay Registers 0-3* on page 3-39
MPMCStaticConfig3	`0x260`	RW	-	`0x0000-0`[a]	See *Static Memory Configuration Registers 0-3* on page 3-33
MPMCStaticWaitWen3	`0x264`	RW	-	`0x0`	See *Static Memory Write Enable Delay Registers 0-3* on page 3-35
MPMCStaticWaitOen3	`0x268`	RW	-	`0x0`	See *Static Memory Output Enable Delay Registers 0-3* on page 3-36
MPMCStaticWaitRd3	`0x26C`	RW	-	`0x1F`	See *Static Memory Read Delay Registers 0-3* on page 3-37
MPMCStaticWaitPage3	`0x270`	RW	-	`0x1F`	See *Static Memory Page Mode Read Delay Registers 0-3* on page 3-37
MPMCStaticWaitWr3	`0x274`	RW	-	`0x1F`	See *Static Memory Write Delay Registers 0-3* on page 3-38
MPMCStaticWaitTurn3	`0x278`	RW	-	`0xF`	See *Static Memory Turnaround Delay Registers 0-3* on page 3-39
MPMCNANDControl	`0x300`	RW	-	`0x00010000`	See *NAND Memory Control Vector Register* on page 3-40
MPMCNANDAddress1	`0x304`	RW	-	`0x00000000`	See *NAND Memory Address Vectors 1-4 Register* on page 3-42
MPMCNANDAddress2	`0x308`	RW	-	`0x00`	See *NAND Memory Address Vector 5 Register* on page 3-43
MPMCNANDTiming1	`0x320`	RW	-	`0x1FFFFFF`	See *NAND Memory Timing Value 1 Register* on page 3-44
MPMCNANDTiming1	`0x324`	RW	-	`0x1F71F1F`	See *NAND Memory Timing Value 2 Register* on page 3-46
MPMCNANDStatus	`0x328`	RO	-	`0x00`	See *NAND Status Information Register* on page 3-47

Table 3-1 MPMC register summary (continued)

Name	Offset from base	Type	Reset HRESETn	Reset nPOR	Description
MPMCAHBControl0	`0x400`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus0	`0x404`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut0	`0x408`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51
MPMCAHBControl1	`0x420`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus1	`0x424`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut1	`0x428`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51
MPMCAHBControl2	`0x440`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus2	`0x444`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut2	`0x448`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51
MPMCAHBControl3	`0x460`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus3	`0x464`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut3	`0x468`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51
MPMCAHBControl4	`0x480`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus4	`0x484`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut4	`0x488`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51

Table 3-1 MPMC register summary (continued)

Name	Offset from base	Type	Reset HRESETn	Reset nPOR	Description
MPMCAHBControl5	`0x4A0`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus5	`0x4A4`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut5	`0x4A8`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51
MPMCAHBControl6	`0x4C0`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus6	`0x4C4`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut6	`0x4C8`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51
MPMCAHBControl7	`0x4E0`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus7	`0x4E4`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut7	`0x4E8`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51
MPMCAHBControl8	`0x500`	RW	`0x0`	-	See *AHB Control Registers 0-8* on page 3-49
MPMCAHBStatus8	`0x504`	RO	`0x0`	-	See *AHB Status Registers 0-8* on page 3-50
MPMCAHBTimeOut8	`0x508`	RW	`0x000`	-	See *AHB Timeout Registers 0-8* on page 3-51
MPMCPeriphID4	`0xFD0`	RO	`0x19`	`0x19`	See *Additional Peripheral Identification Register 4* on page 3-52
MPMCPeriphID5	`0xFD4`	RO	`0x00`	`0x00`	See *Additional Peripheral Identification Register 5-7* on page 3-53
MPMCPeriphID6	`0xFD8`	RO	`0x00`	`0x00`	See *Additional Peripheral Identification Register 5-7* on page 3-53

Table 3-1 MPMC register summary (continued)

Name	Offset from base	Type	Reset HRESETn	Reset nPOR	Description
MPMCPeriphID7	`0xFDC`	RO	`0x00`	`0x00`	See *Additional Peripheral Identification Register 5-7* on page 3-53
MPMCPeriphID0	`0xFE0`	RO	`0x76`	`0x76`	See *Peripheral Identification Register 0* on page 3-54
MPMCPeriphID1	`0xFE4`	RO	`0x11`	`0x11`	See *Peripheral Identification Register 1* on page 3-55
MPMCPeriphID2	`0xFE8`	RO	`0x14`	0x14[b]	See *Peripheral Identification Register 2* on page 3-55
MPMCPeriphID3	`0xFEC`	RO	`0x99`	`0x99`	See *Peripheral Identification Register 3* on page 3-56
MPMCPCellID0	`0xFF0`	RO	`0x0D`	`0x0D`	See *PrimeCell Identification Registers 0-3* on page 3-56
MPMCPCellID1	`0xFF4`	RO	`0xF0`	`0xF0`	See *PrimeCell Identification Registers 0-3* on page 3-56
MPMCPCellID2	`0xFF8`	RO	`0x05`	`0x05`	See *PrimeCell Identification Registers 0-3* on page 3-56
MPMCPCellID3	`0xFFC`	RO	`0xB1`	`0xB1`	See *PrimeCell Identification Registers 0-3* on page 3-56

a. Tie-off dependent.
b. Revision dependent.

3.3 Register descriptions

This section describes the MPMC registers with the exception of the test registers which are described in Chapter 4 *Programmer's Model for Test*. Table 3-1 on page 3-3 provides cross references to individual registers.

3.3.1 Control Register

The MPMCControl Register is a two-bit, read/write register that controls the memory controller operation. The register fields can only be altered in idle state, when no external memory accesses are being performed. This register is accessed with zero wait states. Figure 3-1 shows the register bit assignments.

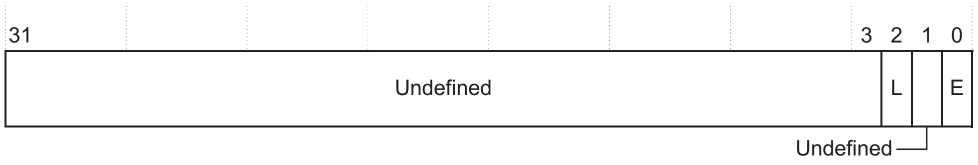


Figure 3-1 MPMCControl Register bit assignments

Table 3-2 lists the register bit assignments.

Table 3-2 MPMCControl Register bit assignments

Bits	Name	Description
[31:3]	-	Read undefined. Write as zero.

Table 3-2 MPMCControl Register bit assignments (continued)

Bits	Name	Description
[2]	Low-power mode (L)	Indicates normal or low-power mode: 0 = normal mode (reset value on **nPOR** and **HRESETn**) 1 = low-power mode. Entering low-power mode reduces memory controller power consumption. Dynamic memory is refreshed as necessary. The memory controller returns to normal functional mode by clearing the low-power mode bit (L), or by AHB, or power-on reset. You must only modify this bit when the MPMC is in idle state.[a][b]
[1]	-	Read undefined. Write as zero.
[0]	MP Enable (E)	Indicates if the MPMC is enabled or disabled: 0 = disabled 1 = enabled (reset value on **nPOR** and **HRESETn**). Disabling the MPMC reduces power consumption. When the memory controller is disabled the memory is not refreshed. The memory controller is enabled by setting the enable bit, or by system, or power-on reset. The MPMC produces an ERROR response (on **HRESP**) to external memory accesses when this bit is LOW. You must only modify this bit when the MPMC is in idle state.[a][b]

a. The external memory cannot be accessed in low-power and/or disable states. If a memory access is performed an error response is generated.

b. The memory controller AHB register programming port can be accessed normally. The MPMC registers can be programmed in low-power and disabled state.

3.3.2 Status Register

The four-bit, read-only MPMCStatus Register provides MPMC status information. This register is accessed with zero wait states. Figure 3-2 shows the register bit assignments.

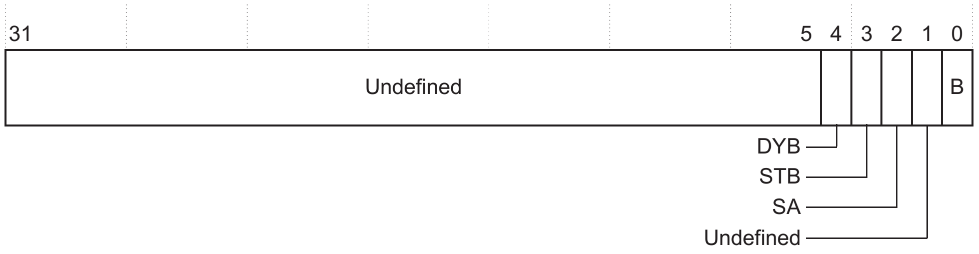


Figure 3-2 MPMCStatus Register bit assignments

Table 3-3 lists the register bit assignments.

Table 3-3 MPMCStatus Register bit assignments

Bits	Name	Description
[31:5]	-	Read undefined.
[4]	DMC Busy (DYB)	This read-only bit is used to ensure that the memory controller enters the self-refresh mode cleanly: 0 = dynamic memory controller is idle 1 = dynamic memory controller is busy performing memory transactions, commands, auto-refresh cycles, or is in self-refresh mode (reset value on **nPOR** and **HRESETn**).
[3]	SMC Busy (STB)	This read-only bit is used to ensure that the memory controller enters the low-power or disabled mode cleanly: 0 = static memory controller is idle (reset value on **nPOR** and **HRESETn**) 1 = static memory controller is busy performing memory transactions.
[2]	Self-refresh acknowledge, **SREFACK, SA**	This read-only bit indicates the operating mode of the dynamic memory controller: 0 = normal mode 1 = self-refresh mode (reset value on **nPOR**).
[1]	-	Read undefined.
[0]	Busy (B)	This read-only bit is used to ensure that the memory controller enters the low-power or disabled mode cleanly. This is a combination of the status of the static, dynamic, and NAND flash memory interfaces. It shows when the NAND flash interface is busy, but not whether a data transfer is currently in progress. The MPMCNANDStatus register NDTX bit must be checked before entering low-power or disabled mode to ensure that a NAND flash access has ended: 0 = MPMC is idle 1 = MPMC is busy performing memory transactions, commands, auto-refresh cycles, or is in self-refresh mode (reset value on **nPOR** and **HRESETn**).

3.3.3 Configuration Register

The one-bit, read/write, MPMCConfig Register configures the operation of the memory controller. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. This register is accessed with one wait state. Figure 3-3 on page 3-13 shows the register bit assignments.

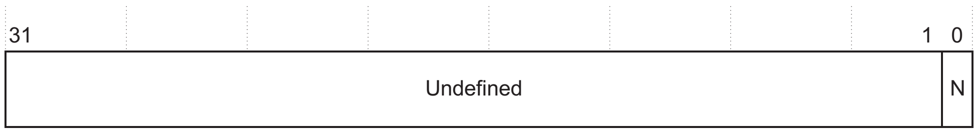


Figure 3-3 MPMCConfig Register bit assignments

Table 3-4 lists the register bit assignments..

Table 3-4 MPMCConfig Register bit assignments

Bits	Name	Description
[31:1]	-	Read undefined. Write as zero.
[0]	Endian mode (N)	Endian mode: 0 = little-endian mode (reset value on **nPOR**) 1 = big-endian mode. The value of the endian bit on **nPOR** is determined by **MPMCBIGENDIAN**. This value can be overridden by software. This field is unaffected by **HRESETn**.[a][b]

a. The value of the **MPMCBIGENDIAN** signal is not reflected in this field. This field reflects the last value that was written into it.

b. Setting little-endian mode in this register has no effect on 64-bit ports which are hardwired as mixed-endian by tying their **MPMCBSTRBENx** signal HIGH. Setting big-endian in this register forces all ports to big-endian mode, irrespective of the state of the **MPMCBSTBRENx** signal.

3.3.4 Dynamic Memory Control Register

The 14-bit, read/write, MPMCDynamicControl Register is used to control dynamic memory operation. The control bits can be altered during normal operation. This register is accessed with zero wait states. Figure 3-4 on page 3-14 shows the register bit assignments.

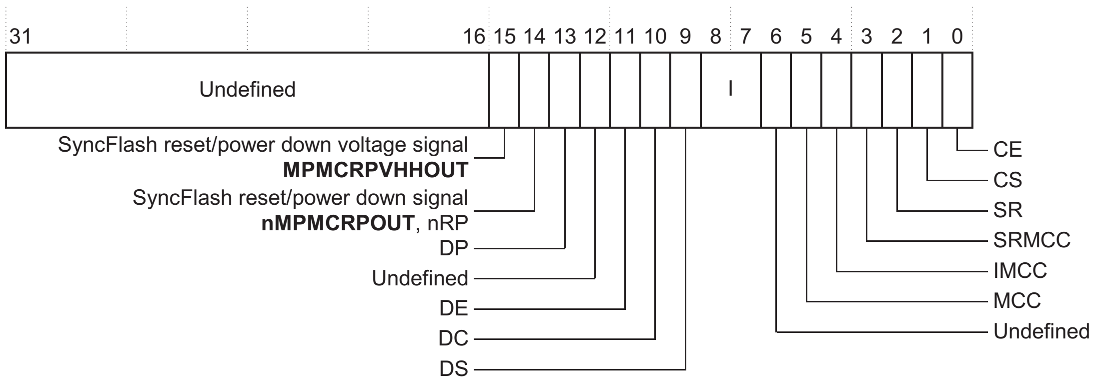

31
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Undefined
I
SyncFlash reset/power down voltage signal MPMCRPVHHOUT
SyncFlash reset/power down signal nMPMCRPOUT, nRP
DP
Undefined
DE
DC
DS
CE
CS
SR
SRMCC
IMCC
MCC
Undefined

Figure 3-4 MPMCDynamicControl Register bit assignments

Table 3-5 lists the register bit assignments.

Table 3-5 MPMCDynamicControl Register bit assignments

Bits	Name	Description
[31:16]	-	Read undefined. Write as zero.
[15]	SyncFlash reset/power down voltage signal, **MPMCRPVHHOUT**	0 = normal voltage (reset value on **nPOR**) 1 = set **MPMCRPOUT** high voltage. This output can be used externally in conjunction with the **nMPMCRPOUT** signal to indicate a high output voltage, see Table 3-6 on page 3-16.
[14]	SyncFlash reset/power down signal, **nMPMCRPOUT** (nRP)	0 = **nMPMCRPOUT** signal LOW (reset value on **nPOR**) 1 = set **nMPMCRPOUT** signal HIGH.
[13]	Low-power SDRAM deep-sleep mode (DP)	0 = normal operation (reset value on **nPOR**) 1 = enter deep power down mode.
[12]	-	Read undefined. Write as zero.
[11]	DLL enable (DE)	Enable DLL hand shaking (**MPMCDLLCALIBREQ**) during auto-refresh cycles: 0 = DLL hand shaking disabled (reset value on **nPOR**) 1 = DLL hand shaking enabled.
[10]	DLL calibrate (DC)	Software control of the DLL hand shaking (**MPMCDLLCALIBREQ**) for initial DLL calibration: 0 = DLL hand shaking disabled (reset value on **nPOR**) 1 = DLL hand shaking enabled.
[9]	DLL status (DS)	Indicates the value of the **MPMCDLLCALIBACK** signal for software control of DLL hand shaking: 0 = DLL calibrating (reset value on **nPOR**) 1 = DLL calibration completed.

Table 3-5 MPMCDynamicControl Register bit assignments (continued)

Bits	Name	Description
[8:7]	SDRAM initialization (I)	00 = issue SDRAM NORMAL operation command (reset value on **nPOR**) 01 = issue SDRAM MODE command 10 = issue SDRAM PALL (precharge all) command 11 = issue SDRAM NOP (no operation) command.
[6]	-	Read undefined. Write as zero.
[5]	Memory clock control (MCC)	0 = **MPMCCLKOUT** enabled (reset value on **nPOR**) 1 = **MPMCCLKOUT** disabled.[a]
[4]	Inverted memory clock control (IMCC)	0 = **nMPMCCLKOUT** enabled (reset value on **nPOR**) 1 = **nMPMCCLKOUT** disabled.[b]
[3]	Self-refresh clock control (SRMCC)	Self-refresh clock control: 0 = **MPMCCLKOUT** and **nMPMCCLKOUT** stop during self-refresh mode. 1 = **MPMCCLKOUT** and **nMPMCCLKOUT** run continuously (reset value on **nPOR**).[c]
[2]	Self-refresh request, **MPMCSREFREQ** (SR)	0 = normal mode 1 = enter self-refresh mode (reset value on **nPOR**). By writing 1 to this bit self-refresh mode can be entered under software control. Writing 0 to this bit returns the MPMC to normal mode. The self-refresh acknowledge bit in the MPMCStatus Register must be polled to determine the current operating mode of the MPMC.
[1]	Dynamic memory clock control (CS)	0 = **MPMCCLKOUT** stops when all SDRAMS are idle and during self-refresh mode1 = **MPMCCLKOUT** runs continuously (reset value on **nPOR**). Programming the dynamic memory clock enable HIGH (CE=1) and dynamic memory clock control LOW (CS =0) results in unpredictable behavior.[d]
[0]	Dynamic memory clock enable (CE)	0 = clock enable of idle devices are deasserted to save power (reset value on **nPOR**) 1 = all clock enables are driven HIGH continuously.[e]

a. Disabling **MPMCCLKOUT** can be performed if there are no SDRAM memory transactions. When enabled this field can be used in conjunction with the dynamic memory clock control (CS) field.
b. Disabling **nMPMCCLKOUT** can be performed if there are no DDR-SDRAM memory transactions that require a differential clock. When enabled this field can be used in conjunction with the dynamic memory clock control (CS) field.
c. This functionality is supported by SDR-SDRAM and DDR-SDRAM.
d. This functionality is only supported by SDR-SDRAM.
e. Clock enable must be HIGH during SDRAM initialization.

Table 3-6 on page 3-16 shows the output voltage settings for different combinations of bits [15:14].

A block external to the MPMC can use the values of the **nMPMCRPOUT** and **MPMCRPVHHOUT** signals to generate the required voltage settings for the Micron SyncFlash reset signal. Some systems do not have an 8V output, or do not require the functionality to raise **nRP** to 8V. In these cases **nMPMCRPVHHOUT** can be ignored.

Table 3-6 Output voltage settings

nMPMCRPOUT	MPMCRPVHHOUT	Output
0	0	0V
0	1	0V
1	0	3V
1	1	8V

3.3.5 Dynamic Memory Refresh Timer Register

The 11-bit, read/write, MPMCDynamicRefresh Register configures dynamic memory operation. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. However, these control bits can, if necessary be altered during normal operation. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-5 shows the register bit assignments.

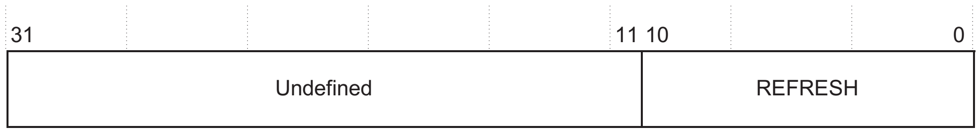


Figure 3-5 MPMCDynamicRefresh Register bit assignments

Table 3-7 lists the register bit assignments.

Table 3-7 MPMCDynamicRefresh Register bit assignments

Bits	Name	Description
[31:11]	-	Read undefined. Write as zero.
[10:0]	Refresh timer (REFRESH)	`0x0` = refresh disabled (reset value on **nPOR**) `0x1` 1(x16) = 16 **HCLK** cycles between SDRAM refresh cycles `0x8` 8(x16) = 128 **HCLK** cycles between SDRAM refresh cycles `0x1`-`0x7FF` n(x16) = 16n **HCLK** cycles between SDRAM refresh cycles.

For example, for the refresh period of 16µs, and an **HCLK** frequency of 50MHz, the following value must be programmed into this register:

$$\frac{16 \times 10^{-6} \times 50 \times 10^{6}}{16} = 50 \text{ or } \texttt{0x32}$$

Note

The refresh cycles are evenly distributed. However, there might be slight variations when the auto-refresh command is issued depending on the status of the memory controller.

3.3.6 Dynamic Memory Read Configuration Register

The six-bit, read/write, MPMCDynamicReadConfig Register enables you to configure the dynamic memory read strategy. This register must only be modified during system initialization. This register is accessed with one wait state.

Figure 3-6 shows the register bit assignments.

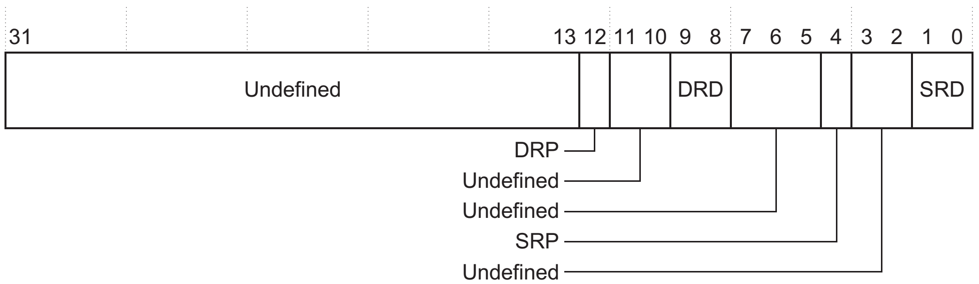


Figure 3-6 MPMCDynamicReadConfig Register bit assignments

Table 3-8 lists the register bit assignments.

Table 3-8 MPMCDynamicReadConfig Register bit assignments

Bits	Name	Description
[31:13]	-	Read undefined. Write as zero.
[12]	DDR-SDRAM read data capture polarity (DRP)	0 = data capture on the negative edge of **HCLK**. 1 = data capture on the positive edge of **HCLK**.[a]
[11:10]	-	Read undefined. Write as zero.
[9:8]	DDR-SDRAM read data strategy (DRD)	00 = clock out delayed strategy, using **MPMCCLKOUT** (command not delayed, clock out delayed). 01 = command delayed strategy, using **MPMCCLKDELAY** (command delayed, clock out not delayed). 10 = command delayed strategy plus one clock cycle, using **MPMCCLKDELAY** (command delayed, clock out not delayed). 11 = command delayed strategy plus two clock cycles, using **MPMCCLKDELAY** (command delayed, clock out not delayed).[b]
[7:5]	-	Read undefined. Write as zero.
[4]	SDR-SDRAM read data capture polarity (SRP)	0 = data capture on the negative edge of **HCLK**. 1 = data capture on the positive edge of **HCLK**.[c]
[3:2]	-	Read undefined. Write as zero.
[1:0]	SDR-SDRAM read data strategy (SRD)	00 = clock out delayed strategy, using **MPMCCLKOUT** (command not delayed, clock out delayed). 01 = command delayed strategy, using **MPMCCLKDELAY** (command delayed, clock out not delayed). 10 = command delayed strategy plus one clock cycle, using **MPMCCLKDELAY** (command delayed, clock out not delayed). 11 = command delayed strategy plus two clock cycles, using **MPMCCLKDELAY** (command delayed, clock out not delayed).[d]

a. The value of the DRP field on **nPOR** is determined by the **MPMCDYDDRPOL** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.

b. The value of the DRD field on **nPOR** is determined by the **MPMCDYDDRDLY[1:0]** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.

c. The value of the SRP field on **nPOR** is determined by the **MPMCDYSDRPOL** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.

d. The value of the SRD field on **nPOR** is determined by the **MPMCDYSDRDLY[1:0]** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.

3.3.7 Dynamic Memory Precharge Command Period Register

The four-bit, read/write, MPMCDynamictRP Register enables you to program the precharge command period, t_{RP}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RP}. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-7 shows the register bit assignments.

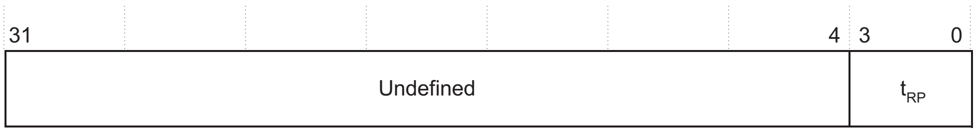


Figure 3-7 MPMCDynamictRP Register bit assignments

Table 3-9 lists the register bit assignments.

Table 3-9 MPMCDynamictRP Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Precharge command period (t_{RP})	`0x0`-`0xE` = n + 1 clock cycles `0xF` = 16 clock cycles (reset value on **nPOR**)

3.3.8 Dynamic Memory Active To Precharge Command Period Register

The four-bit, read/write, MPMCDynamictRAS Register enables you to program the active to precharge command period, t_{RAS}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RAS}. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-8 on page 3-20 shows the register bit assignments.

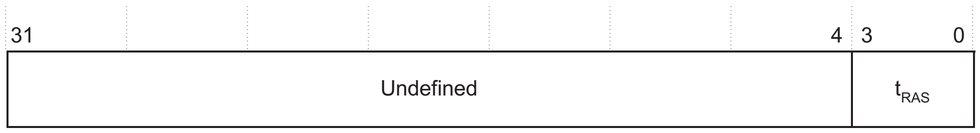


Figure 3-8 MPMCDynamictRAS Register bit assignments

Table 3-10 lists the register bit assignments..

Table 3-10 MPMCDynamictRAS Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Active to precharge command period (t_{RAS})	`0x0`-`0xE` = n+1 clock cycles `0xF` = 16 clock cycles (reset value on **nPOR**)

3.3.9 Dynamic Memory Self-refresh Exit Time Register

The seven-bit, read/write, MPMCDynamictSREX Register enables you to program the self-refresh exit time, t_{SREX}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{SREX}, for devices without this parameter you must use the same value as t_{XSR}. For some DDR-SDRAM data sheets this parameter is known as t_{XSNR}. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-9 shows the register bit assignments.

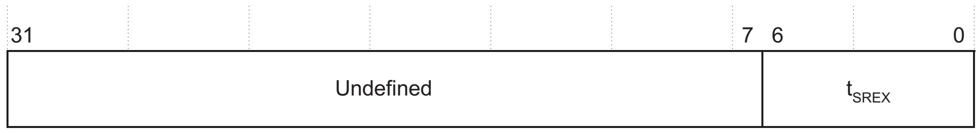


Figure 3-9 MPMCDynamictSREX Register bit assignments

Table 3-11 lists the register bit assignments.

Table 3-11 MPMCDynamictSREX Register bit assignments

Bits	Name	Description
[31:7]	-	Read undefined. Write as zero.
[6:0]	Self-refresh exit time (t_{SREX})	`0x0`-`0x7F` = n+1 clock cycles `0x7F` = 128 clock cycles (reset value on **nPOR**)

3.3.10 Dynamic Memory Write Recovery Time Register

The four-bit, read/write, MPMCDynamictWR Register enables you to program the write recovery time, t_{WR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{WR}, t_{DPL}, t_{RWL}, or t_{RDL}. This register is accessed with one wait state.

——— **Note** ———

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

———————————

Figure 3-10 shows the register bit assignments.

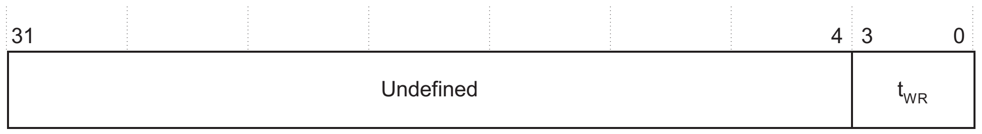


Figure 3-10 MPMCDynamictWR Register bit assignments

Table 3-12 lists the register bit assignments.

Table 3-12 MPMCDynamictWR Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Write recovery time (t_{WR})	`0x0`-`0xE` = n+1 clock cycles `0xF` = 16 clock cycles (reset value on **nPOR**)

3.3.11 Dynamic Memory Active To Active Command Period Register

The five-bit, read/write, MPMCDynamictRC Register enables you to program the active to active command period, t_{RC}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RC}. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-11 shows the register bit assignments.

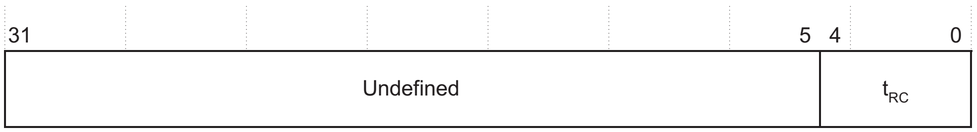


Figure 3-11 MPMCDynamictRC Register bit assignments

Table 3-13 lists the register bit assignments.

Table 3-13 MPMCDynamictRC Register bit assignments

Bits	Name	Description
[31:5]	-	Read undefined. Write as zero.
[4:0]	Active to active command period (t_{RC})	`0x0`-`0x1E` = n+1 clock cycles `0x1F` = 32 clock cycles (reset value on **nPOR**)

3.3.12 Dynamic Memory Auto-refresh Period Register

The five-bit, read/write, MPMCDynamictRFC Register enables you to program the auto-refresh period, and auto-refresh to active command period, t_{RFC}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RFC}, or sometimes as t_{RC}. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-12 shows the register bit assignments.

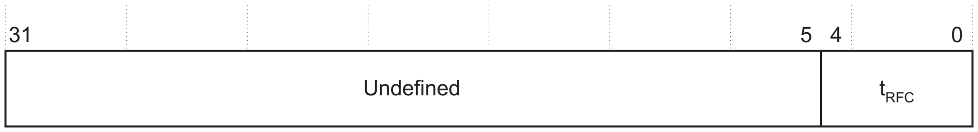


Figure 3-12 MPMCDynamictRFC Register bit assignments

Table 3-14 lists the register bit assignments.

Table 3-14 MPMCDynamictRFC Register bit assignments

Bits	Name	Description
[31:5]	-	Read undefined. Write as zero.
[4:0]	Auto-refresh period and auto-refresh to active command period (t_{RFC})	`0x0`-`0x1E` = n+1 clock cycles `0x1F` = 32 clock cycles (reset value on **nPOR**)

3.3.13 Dynamic Memory Exit Self-refresh Register

The eight-bit, read/write, MPMCDynamictXSR Register enables you to program the exit self-refresh to active command time, t_{XSR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{XSR}, but is sometimes called t_{XSNR} in some DDR-SDRAM data sheets. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-13 on page 3-24 shows the register bit assignments.

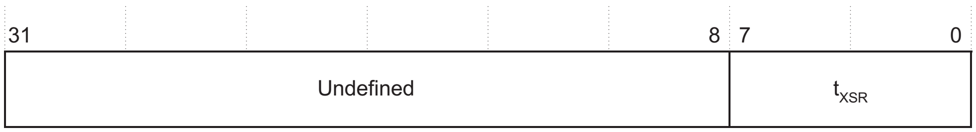


Figure 3-13 MPMCDynamictXSR Register bit assignments

Table 3-15 lists the register bit assignments.

Table 3-15 MPMCDynamictXSR Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined. Write as zero.
[7:0]	Exit self-refresh to active command time (t_{XSR})	`0x0`-`0xFE` = n+1 clock cycles `0xFF` = 256 clock cycles (reset value on **nPOR**)

3.3.14 Dynamic Memory Active Bank A-B Time Register

The four-bit, read/write, MPMCDynamictRRD Register enables you to program the active bank A to active bank B latency, t_{RRD}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{RRD}. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-14 shows the register bit assignments.

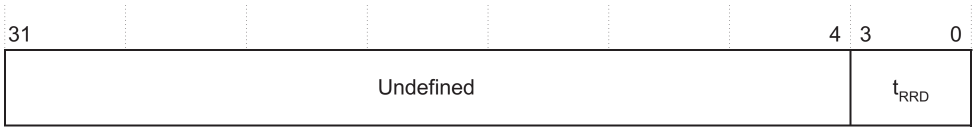


Figure 3-14 MPMCDynamictRRD Register bit assignments

Table 3-16 lists the register bit assignments.

Table 3-16 MPMCDynamictRRD Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Active bank A to active bank B latency (t_{RRD})	`0x0`-`0xE` = n+1 clock cycles `0xF` = 16 clock cycles (reset value on **nPOR**)

3.3.15 Dynamic Memory Load Mode Register

The four-bit, read/write, MPMCDynamictMRD Register enables you to program the load mode register to active command time, t_{MRD}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{MRD} or t_{RSA}. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-15 shows the register bit assignments.

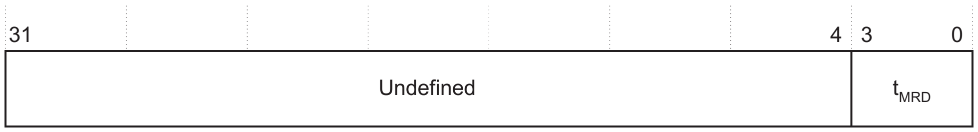


Figure 3-15 MPMCDynamictMRD Register bit assignments

Table 3-17 lists the register bit assignments.

Table 3-17 MPMCDynamictMRD Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Load mode register to active command time (t_{MRD})	`0x0`-`0xE` = n+1 clock cycles `0xF` = 16 clock cycles (reset value on **nPOR**)

3.3.16 Dynamic Memory Last Data In To Read Command Time Register

The four-bit, read/write, MPMCDynamictCDLR Register enables you to program the last data in to read command time, t_{CDLR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. This value is normally found in SDRAM data sheets as t_{CDLR}. This register is accessed with one wait state.

Note

This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Figure 3-16 shows the register bit assignments

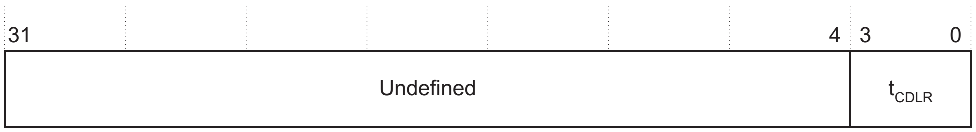


Figure 3-16 MPMCDynamictCDLR Register bit assignments

Table 3-18 lists the register bit assignments.

Table 3-18 MPMCDynamictCDLR Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Last data in to read command time (t_{CDLR})	`0x0`-`0xE` = n+1 clock cycles `0xF` = 16 clock cycles (reset value on **nPOR**)

3.3.17 Static Memory Extended Wait Register

The 10-bit, read/write, MPMCStaticExtendedWait Register is used to time long static memory read and write transfers (that are longer than can be supported by the MPMCStaticWaitRd0-3 or, MPMCStaticWaitWr0-3 Registers) when the EW bit of the MPMCStaticConfig Registers is enabled. There is only a single MPMCStaticExtendedWait Register. This is used by the relevant static memory chip select if the appropriate *Extended Wait* (EW) bit in the MPMCStaticConfig Register is set. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. However, if necessary, these control bits can be altered during normal operation. This register is accessed with one wait state.

Figure 3-17 shows the register bit assignments.

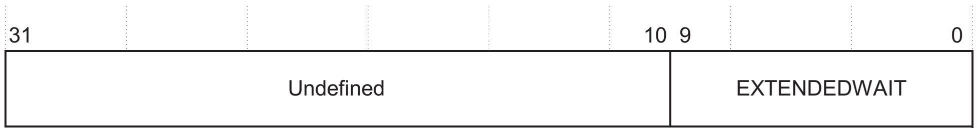


Figure 3-17 MPMCStaticExtendedWait Register bit assignments

Table 3-19 lists the register bit assignments.

Table 3-19 MPMCStaticExtendedWait Register bit assignments

Bits	Name	Description
[31:10]	-	Read undefined. Write as zero.
[9:0]	External wait time out (EXTENDEDWAIT)	`0x0` = 16 clock cycles (reset value on **nPOR**) `0x1`-`0x3FF` = (n+1) x16 clock cycles

For example, for a static memory read or write transfer time of 16μs, and an **HCLK** frequency of 50MHz, the following value must be programmed into this register:

$$\frac{16 \times 10^{-6} \times 50 \times 10^{6}}{16} - 1 = 49$$

3.3.18 Dynamic Memory Configuration Registers 0-3

The 13-bit, read/write, MPMCDynamicConfig0-3 Registers enable you to program the configuration information for the relevant dynamic memory chip select. These registers are normally only modified during system initialization. These registers are accessed with one wait state.

Figure 3-18 on page 3-28 shows the register bit assignments

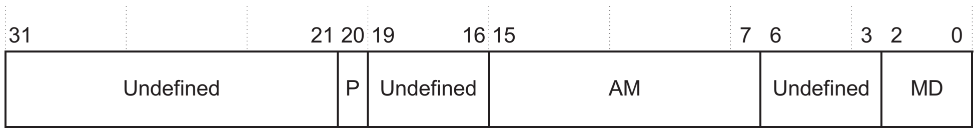


Figure 3-18 MPMCDynamicConfig0-3 Register bit assignments

Table 3-20 lists the register bit assignments.

Table 3-20 MPMCDynamicConfig0-3 Register bit assignments

Bits	Name	Description
[31:21]	-	Read undefined. Write as zero.
[20]	Write protect (P)	0 = writes not protected (reset value on **nPOR**) 1 = write protected.
[19:16]	-	Read undefined. Write as zero.
[15:7]	Address mapping (AM)	`0x000` = 16Mb (2Mx8) 2 banks, row length = 11, column length = 9 (reset value for chip selects 4, 6, and 7, on **nPOR**) For `0x001-1FF`, see Table 3-21 on page 3-29. The value of the chip select 5 address mapping field on **nPOR** is determined by **MPMCDYCS5ADRMAP[8:0]**. This value can be overridden by software. This field is unaffected by **HRESETn**.
[6:3]	-	Read undefined. Write as zero.
[2:0]	Memory device (MD)	000 = SDR-SDRAM (reset value for chip selects 4, 6, and 7 on **nPOR**)001 = SDR Micron SyncFlash010 = low-power SDR-SDRAM011 = SDR Micron V-SyncFlash100 = DDR-SDRAM101 = DDR Micron SyncFlash110 = low-power DDR-SDRAM111 = DDR Micron V-SyncFlash. The value of the chip select 5 memory device field on **nPOR** is determined by **MPMCDYCS5DEVICE[2:0]**. This value can be overridden by software. This field is unaffected by **HRESETn**.

Note

DDR-SDRAM device chip selects must only have a 16 or 32-bit wide databus.

Table 3-21 lists all combinations of external bus width, mapping strategy, device capacity, and device width which are supported by MPMC. Address mappings that are not shown in Table 3-21 are reserved. Bits [15:7] are allocated as follows:

Bit[15:14]	External bus width (chip select width) which can be 16-bit (00), 32-bit (01), or 64-bit (10).
Bit[13:12]	Mapping strategy, which can be high performance (row, bank, column (RBC) (00)), low power (bank, row, column (BRC) (01)), or V-SyncFlash (bank, segment, row, segment, column (BSRSC) (10)).
Bit[11:9]	Device capacity, which can be 16Mb (000), 64Mb (001), 128Mb (010), 256Mb (011), or 512Mb (100).
Bits [8:7]	Device width, which can be 8-bit (00), 16-bit (01), or 32-bit (10).

Table 3-21 Address mapping

[15:14]	[13:12]	[11:9]	[8:7]	Description
16-bit external bus high-performance SDRAM address mapping (Row, Bank, Column)				
00	00	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
00	00	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
00	00	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
00	00	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
00	00	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
00	00	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
00	00	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
00	00	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
00	00	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
00	00	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
16-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
00	01	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
00	01	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
00	01	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
00	01	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8

Table 3-21 Address mapping (continued)

[15:14]	[13:12]	[11:9]	[8:7]	Description
00	01	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
00	01	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
00	01	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
00	01	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
00	01	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
00	01	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
32-bit external bus high-performance SDRAM address mapping (Row, Bank, Column)				
01	00	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
01	00	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
01	00	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
01	00	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
01	00	001	10	64Mb (2Mx32), 4 banks, row length = 11, column length = 8
01	00	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
01	00	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
01	00	010	10	128Mb (4Mx32), 4 banks, row length = 12, column length = 8
01	00	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
01	00	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
01	00	011	10	256Mb (8Mx32), 4 banks, row length = 13, column length = 8
01	00	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
01	00	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
32-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
01	01	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
01	01	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
01	01	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
01	01	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8

Table 3-21 Address mapping (continued)

[15:14]	[13:12]	[11:9]	[8:7]	Description
01	01	001	10	64Mb (2Mx32), 4 banks, row length = 11, column length = 8
01	01	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
01	01	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
01	01	010	10	128Mb (4Mx32), 4 banks, row length = 12, column length = 8
01	01	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
01	01	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
01	01	011	10	256Mb (8Mx32), 4 banks, row length = 13, column length = 8
01	01	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
01	01	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
64-bit external bus high-performance SDRAM address mapping (Bank, Row, Column)				
10	00	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
10	00	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
10	00	001	10	64Mb (2Mx32), 4 banks, row length = 11, column length = 8
10	00	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
10	00	010	10	128Mb (4Mx32), 4 banks, row length = 12, column length = 8
10	00	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
10	00	011	10	256Mb (8Mx32), 4 banks, row length = 13, column length = 8
10	00	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
64-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
10	01	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
10	01	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
10	01	001	10	64Mb (2Mx32), 4 banks, row length = 11, column length = 8
10	01	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
10	01	010	10	128Mb (4Mx32), 4 banks, row length = 12, column length = 8

Table 3-21 Address mapping (continued)

[15:14]	[13:12]	[11:9]	[8:7]	Description
10	01	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
10	01	011	10	256Mb (8Mx32), 4 banks, row length = 13, column length = 8
10	01	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10

A chip select can be connected to a single memory device. In this case the chip select databus width is the same as the device width. Alternatively the chip select can be connected to several external devices. In this case the chip select databus width is the sum of the memory device databus widths.

For example, for a chip select connected to:

- a 64-bit wide memory device, choose a 64-bit wide address mapping
- a 32-bit wide memory device, choose a 32-bit wide address mapping
- a 16-bit wide memory device, choose a 16-bit wide address mapping
- four x 16-bit wide memory devices, choose a 64-bit wide address mapping
- four x 8-bit wide memory devices, choose a 32-bit wide address mapping
- two x 8-bit wide memory devices, choose a 16-bit wide address mapping.

3.3.19 Dynamic Memory RAS and CAS Delay Registers 0-3

The eight-bit, read/write, MPMCDynamicRasCas0-3 Registers enable you to program the RAS and CAS latencies for the relevant dynamic memory. These registers must only be modified during system initialization. This value is normally found in SDRAM data sheets as t_{RAS}. These registers are accessed with one wait state.

Figure 3-19 shows the register bit assignments.

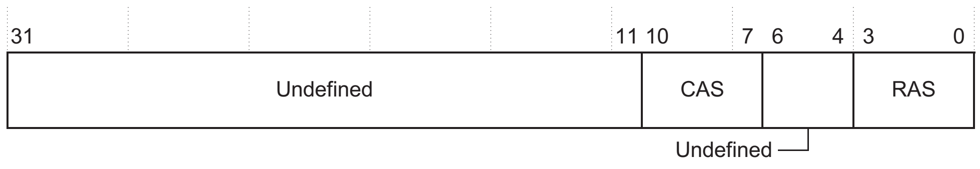


Figure 3-19 MPMCDynamicRasCas0-3 Register bit assignments

Table 3-22 lists the register bit assignments.

Table 3-22 MPMCDynamicRasCas0-3 Register bit assignments

Bits	Name	Description
[31:11]	-	Read undefined. Write as zero.
[10:7]	CAS latency (CAS)	`0x0` = reserved`0x1` = 0.5 **HCLK** cycles`0x2` = 1 **HCLK** cycle`0x3` = 1.5 **HCLK** cycles`0x4`-`0xD` = n/2 **HCLK** cycles (2-6.5)`0xE` = 7 **HCLK** cycles`0xF` = 7.5 **HCLK** cycles (reset value for chip selects 4, 6, and 7, on **nPOR**). The value of the upper three bits of the CAS field for dynamic bank one (chip select 5) on **nPOR** is determined by **MPMCDYCS5CASDLY[2:0]**. This value can be overridden by software. This field is unaffected by **HRESETn**.
[6:4]	-	Read undefined. Write as zero.
[3:0]	RAS latency (active to read or write delay) (RAS)	`0x0` = reserved`0x1`-`0xE` = n **HCLK** cycles (reset value on **nPOR** = `0x3`)`0xF` = 15 **HCLK** cycles.

3.3.20 Static Memory Configuration Registers 0-3

The eight-bit, read/write, MPMCStaticConfig0-3 Registers are used to configure the static memory configuration. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle and then entering low-power or disabled mode. These registers are accessed with one wait state.

Figure 3-20 shows the register bit assignments.

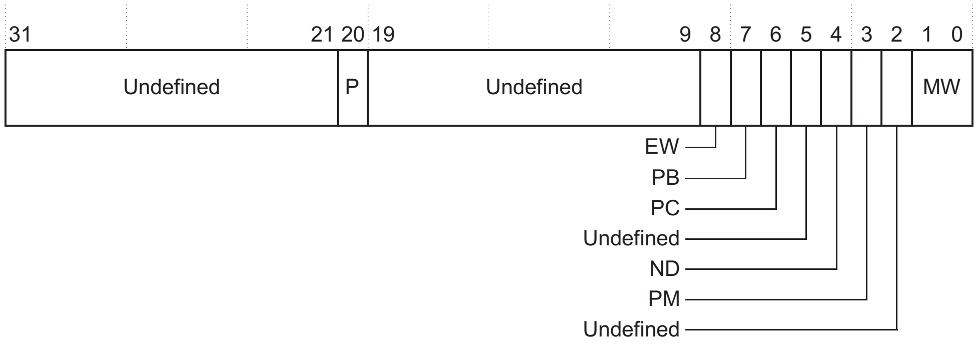


Figure 3-20 MPMCStaticConfig0-3 Register bit assignments

Table 3-23 lists the register bit assignments.

Table 3-23 MPMCStaticConfig0-3 Register bit assignments

Bits	Name	Description
[31:21]	-	Read undefined. Write as zero.
[20]	Write protect (P)	0 = writes not protected (reset value on **nPOR**) 1 = write protected.
[19:9]	-	Read undefined. Write as zero.
[8]	Extended wait (EW)	0 = Extended wait disabled (reset value on **nPOR**) 1 = Extended wait enabled. Extended wait (EW) uses the MPMCStaticExtendedWait Register to time both the read and write transfers rather than the MPMCStaticWaitRd and MPMCStaticWaitWr Registers. This enables much longer transactions.[a]
[7]	Byte lane state (PB)	0 = For reads all the bits in **nMPMCBLSOUT[3:0]** are HIGH. For writes the respective active bits in **nMPMCBLSOUT[3:0]** are LOW (reset value for chip select 0, 2, and 3 on **nPOR**). 1 = For reads the respective active bits in **nMPMCBLSOUT[3:0]** are LOW. For writes the respective active bits in **nMPMCBLSOUT[3:0]** are LOW. The value of the chip select 1 byte lane state field on **nPOR** is determined by **MPMCSTCS1PB**. This value can be overridden by software. This field is unaffected by **HRESETn**.[b]
[6]	Chip select polarity (PC)	0 = active LOW chip select 1 = active HIGH chip select. The value of the chip select polarity on **nPOR** is determined by the relevant **MPMCSTCSxPOL** signal. This value can be overridden by software. This field is unaffected by **HRESETn**.[c]
[5]	-	Read undefined. Write as zero.
[4]	NAND flash (ND)	0 = standard SRAM/ROM/flash device (reset value on **nPOR**) 1 = NAND flash device. Sets the type of memory device found on the chip select, either a standard SRAM interface device (SRAM, ROM, flash), or a NAND flash device which has a different interface. If a NAND device is selected, the NAND control registers must be used to control memory accesses.

Table 3-23 MPMCStaticConfig0-3 Register bit assignments (continued)

Bits	Name	Description
[3]	Page mode (PM)	0 = disabled (reset value on **nPOR**) 1 = asynchronous page mode enabled (page length four). In page mode the MPMC can burst up to four external accesses. Therefore devices with asynchronous page mode burst four or higher devices are supported. Asynchronous page mode burst two devices are not supported and must be accessed normally.
[2]	-	Read undefined. Write as zero.
[1:0]	Memory width (MW)	00 = 8-bit (reset value for chip select 0, 2, and 3 on **nPOR**). 01 = 16-bit 10 = 32-bit 11 = reserved. The value of the chip select 1 memory width field on **nPOR** is determined by **MPMCSTCS1MW[1:0]**. This value can be overridden by software. This field is unaffected by **HRESETn**.

a. Extended wait and page mode cannot be selected simultaneously.
b. For chip select 1 the value of the **MPMCSTCS1PB** signal is reflected in this field. When programmed this register reflects the last value that is written into it.
c. The value of the relevant **MPMCSTCSxPOL** signal is reflected in this field. When programmed this register reflects the last value that is written into it.

3.3.21 Static Memory Write Enable Delay Registers 0-3

The four-bit, read/write, MPMCStaticWaitWen0-3 Registers enable you to program the delay from the chip select to the write enable. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. These registers are accessed with one wait state.

Figure 3-21 shows the register bit assignments.

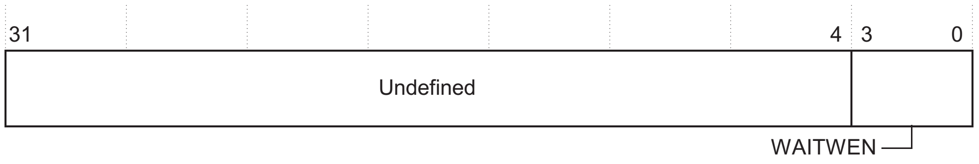


Figure 3-21 MPMCStaticWaitWen0-3 Register bit assignments

Table 3-24 lists the register bit assignments.

Table 3-24 MPMCStaticWaitWen0-3 Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Wait write enable (WAITWEN)	Delay from chip select assertion to write enable: `0x0` = one **HCLK** cycle delay between assertion of chip select and write enable (reset value on **nPOR**) `0x1-0xF` = (n + 1) **HCLK** cycle delay[a]

a. The delay is (WAITWEN +1) x $t_{\mathbf{HCLK}}$.

3.3.22 Static Memory Output Enable Delay Registers 0-3

The four-bit, read/write, MPMCStaticWaitOen0-3 Registers enable you to program the delay from the chip select or address change, whichever is later, to the output enable. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. These registers are accessed with one wait state.

Figure 3-22 shows the register bit assignments.

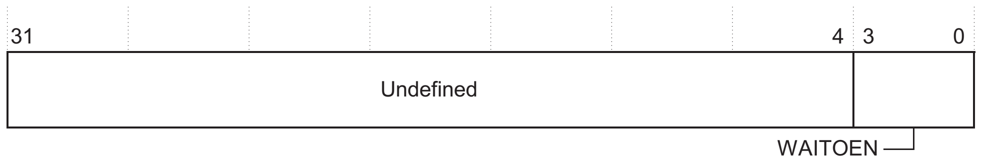


Figure 3-22 MPMCStaticWaitOen0-3 Register bit assignments

Table 3-25 lists the register bit assignments.

Table 3-25 MPMCStaticWaitOen0-3 Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Wait output enable (WAITOEN)	Delay from chip select assertion to output enable: `0x0` = No delay (reset value on **nPOR**) `0x1-0xF` = n cycle delay[a]

a. The delay is WAITOEN x t_{HCLK}.

3.3.23 Static Memory Read Delay Registers 0-3

The five-bit, read/write, MPMCStaticWaitRd0-3 Registers enable you to program the delay from the chip select to the read access. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. These registers are not used if the Extended Wait (EW) bit is enabled in the MPMCStaticConfig0-3 Registers. These registers are accessed with one wait state.

Figure 3-23 shows the register bit assignments.

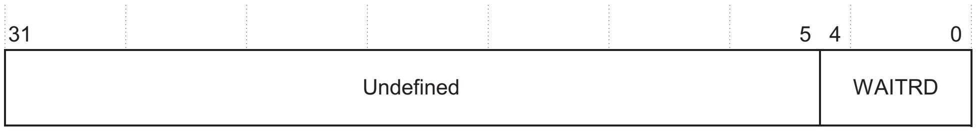


Figure 3-23 MPMCStaticWaitRd0-3 Register bit assignments

Table 3-26 lists the register bit assignments.

Table 3-26 MPMCStaticWaitRd0-3 Register bit assignments

Bits	Name	Description
[31:5]	-	Read undefined. Write as zero.
[4:0]	Nonpage mode read wait states or asynchronous page mode read first access wait state (WAITRD)	Nonpage mode read or asynchronous page mode read, first read only: `0x00-0x1E` = (n + 1) **HCLK** cycles for read accesses[a] `0x1F` = 32 **HCLK** cycles for read accesses (reset value on **nPOR**)

a. For nonsequential reads, the wait state time is (WAITRD + 1) x t_{HCLK}.

3.3.24 Static Memory Page Mode Read Delay Registers 0-3

The five-bit, read/write, MPMCStaticWaitPage0-3 Registers enable you to program the delay for asynchronous page mode sequential accesses. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. These registers are accessed with one wait state.

Figure 3-24 on page 3-38 shows the register bit assignments.

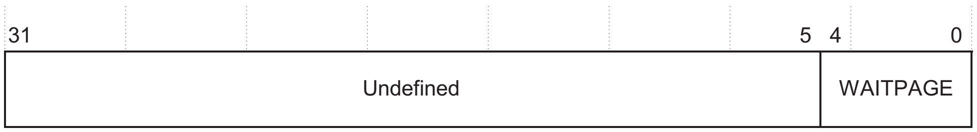


Figure 3-24 MPMCStaticWaitPage0-3 Register bit assignments

Table 3-27 lists the register bit assignments.

Table 3-27 MPMCStaticWaitPage0-3 Register bit assignments

Bits	Name	Description
[31:5]	-	Read undefined. Write as zero.
[4:0]	Asynchronous page mode read after the first read wait states (WAITPAGE)	Number of wait states for asynchronous page mode read accesses after the first read: `0x00-0x1E` = (n+ 1) **HCLK** cycle read access time[a] `0x1F` = 32 **HCLK** cycle read access time (reset value on **nPOR**)

a. For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITPAGE + 1) x $t_{\mathbf{HCLK}}$

3.3.25 Static Memory Write Delay Registers 0-3

The five-bit, read/write, MPMCStaticWaitWr0-3 Registers enable you to program the delay from the chip select to the write access. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. These registers are not used if the Extended Wait (EW) bit is enabled in the MPMCStaticConfig0-3 Registers. These registers are accessed with one wait state.

Figure 3-25 shows the register bit assignments.

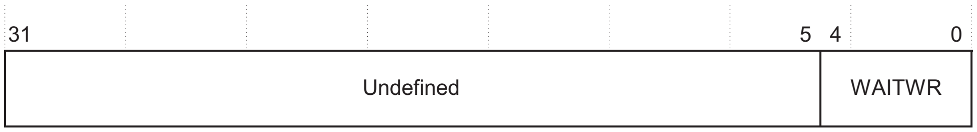


Figure 3-25 MPMCStaticWaitWr0-3 Register bit assignments

Table 3-28 lists the register bit assignments.

Table 3-28 MPMCStaticWaitWr0-3 Register bit assignments

Bits	Name	Description
[31:5]	-	Read undefined. Write as zero.
[4:0]	Write wait states (WAITWR)	SRAM wait state time for write accesses after the first read:`0x00`-`0x1E` = (n + 2) **HCLK** cycle write access time[a]`0x1F` = 33 **HCLK** cycle write access time (reset value on **nPOR**)

a. The wait state time for write accesses after the first read is WAITWR x $t_{\mathbf{HCLK}}$

3.3.26 Static Memory Turnaround Delay Registers 0-3

The four-bit, read/write, MPMCStaticWaitTurn0-3 Registers enable you to program the number of bus turnaround cycles. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. These registers are accessed with one wait state.

Figure 3-26 shows the register bit assigments.

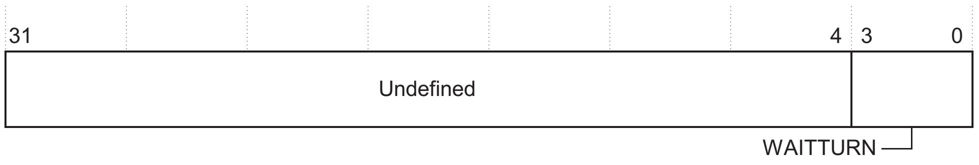


Figure 3-26 MPMCStaticWaitTurn0-3 Register bit assignments

Table 3-29 lists the bit assignments for the MPMCStaticWaitTurn0-3 Registers.

Table 3-29 MPMCStaticWaitTurn0-3 Register bit assignments

Bits	Name	Description
[31:4]	-	Read undefined. Write as zero.
[3:0]	Bus turnaround cycles (WAITTURN)	`0x0`-`0xE` = (n + 1) **HCLK** turnaround cycles[a]`0xF` = 16 **HCLK** turnaround cycles (reset value on **nPOR**)

a. Bus turnaround time is (WAITTURN + 1) x $t_{\mathbf{HCLK}}$

To prevent bus contention on the external memory databus, the WAITTURN field controls the number of bus turnaround cycles added between static memory read and write accesses.

The WAITTURN field also controls the number of turnaround cycles between static memory and dynamic memory accesses.

3.3.27 NAND Memory Control Vector Register

The 28-bit, read/write, MPMCNANDControl Register enables you to program the NAND flash command vector values, and to set other control values for the NAND flash transfer. The register fields can be altered during normal operation, but it is recommended that they are only modified before a NAND access is initiated. This register is accessed with zero wait states.

Figure 3-27 shows the register bit assignments.

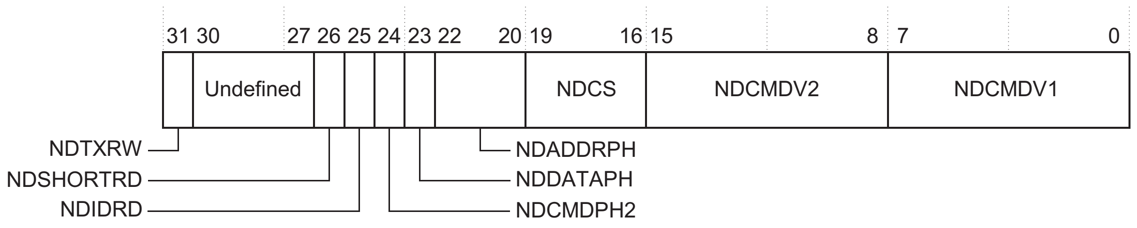


Figure 3-27 MPMCNANDControl Register bit assignments

Table 3-30 lists the bit assignments.

Table 3-30 MPMCNANDControl Register bit assignments

Bits	Name	Description
[31]	Transfer type (NDTXRW)	0 = read transfer (reset value on **nPOR**) 1 = write transfer. Defines the type of transfer that is performed, either a read, or a write. A transfer that performs a data read must be programmed as a read. A transfer that performs a data write or has no data phase, such as a block erase, must be programmed as a write.
[30:27]	-	Read undefined. Write as zero.
[26]	Short read timing (NDSHORTRD)	0 = standard read transfer (reset value on **nPOR**) 1 = short read transfer. Identifies when a read transfer is performed without a check for the ready/busy output status. A standard read transfer checks that the device ready/busy output indicates the device is ready before performing the data phase of the transfer. For a short read transfer, the t_{CLR} timing value is used to time the read transfer delay between the deassertion of the command latch enable, and the assertion of the read enable. Commands such as random page read require this setting because the ready/busy output is not used to control the delay to the data phase. This bit does not have to be set for a Status Read, because this transfer is automatically detected and is performed differently because it does not have an address phase.
[25]	ID read command (NDIDRD)	0 = command is not ID read (reset value on **nPOR**) 1 = read ID command is performed. Identifies that the transfer being performed is reading the device ID, enabling the correct timing to be applied during the transfer (t_{AR1}).
[24]	Second command phase (NDCMDPH2)	0 = no second command phase (reset value on **nPOR**) 1 = second command phase is performed. Controls whether a second command phase is performed at the end of the transfer. Commands such as page program might require a second command phase in the transfer.
[23]	Data phase (NDDATAPH)	0 = no data phase (reset value on **nPOR**) 1 = data phase is performed. Controls whether a data phase is performed during the transfer. Commands such as block erase might not require a data phase in the transfer.

Table 3-30 MPMCNANDControl Register bit assignments (continued)

Bits	Name	Description
[22:20]	Address phase (NDADDRPH)	`0x0` = no address phase (reset value on **nPOR**) `0x1`-`0x5` = n address vectors in address phase `0x6`-`0x7` = reserved. Controls whether an address phase is performed during the transfer, and the number of address vectors in the address phase. Commands such as a status read might not require an address phase in the transfer.
[19:16]	Chip select for transfer (NDCS)	0001 = chip select 0 (reset value on **nPOR**) 0010 = chip select 1 0100 = chip select 2 1000 = chip select 3. Sets the chip select to use for the NAND access. Only one bit of the register can be valid at a time.
[15:8]	Second command vector (NDCMDV2)	`0x00`-`0xFF` = NAND flash second command vector (reset value on **nPOR** is `0x00`). See the applicable specification for the NAND device for the commands supported. This field is programmed for each NAND access that is performed. The second command vector is only used if the NDCMDPH2 bit of the MPMCNANDControl Register is set HIGH.
[7:0]	First command vector (NDCMDV1)	`0x00`-`0xFF` = NAND flash first command vector (reset value on **nPOR** is `0x00`). See the applicable specification for the NAND device for the commands supported. This field is programmed for each NAND access that is performed.

3.3.28 NAND Memory Address Vectors 1-4 Register

The 32-bit, read/write, MPMCNANDAddress1 Register enables you to program the NAND flash address phase values, for vectors 1-4. The register fields can be altered during normal operation, but it is recommended that they are only modified before a NAND access is initiated. This register is accessed with zero wait states.

Figure 3-28 shows the register bit assignments.

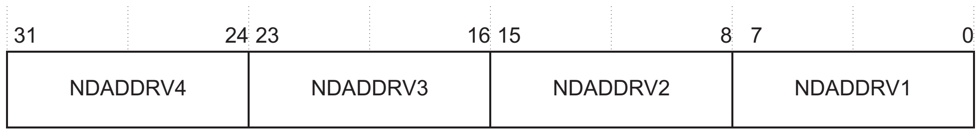


Figure 3-28 MPMCNANDAddress1 Register bit assignments

Table 3-31 lists the register bit assignments.

Table 3-31 MPMCNANDAddress1 Register bit assignments

Bits	Name	Description
[31:24]	Fourth address vector (NDADDRV4)	`0x00-0xFF` = NAND flash fourth address vector (reset value on **nPOR** is `0x0`). This value is only used if the NDADDRPH bits of the MPMCNANDControl Register are set to four or greater.
[23:16]	Third address vector (NDADDRV3)	`0x00-0xFF` = NAND flash third address vector (reset value on **nPOR** is `0x0`). This value is only used if the NDADDRPH bits of the MPMCNANDControl Register are set to three or greater.
[15:8]	Second address vector (NDADDRV2)	`0x00-0xFF` = NAND flash second address vector (reset value on **nPOR** is `0x0`). This value is only used if the NDADDRPH bits of the MPMCNANDControl Register are set to two or greater.
[7:0]	First address vector (NDADDRV1)	`0x00-0xFF` = NAND flash first address vector (reset value on **nPOR** is `0x0`). This value is only used if the NDADDRPH bits of the MPMCNANDControl Register are set to a nonzero value.

3.3.29 NAND Memory Address Vector 5 Register

The eight-bit, read/write, MPMCNANDAddress2 Register enables you to program the NAND flash address phase values, for vector 5. The register field can be altered during normal operation, but it is recommended that it is only modified before a NAND access is initiated. This register is accessed with zero wait states.

Figure 3-29 shows the register bit assignments.

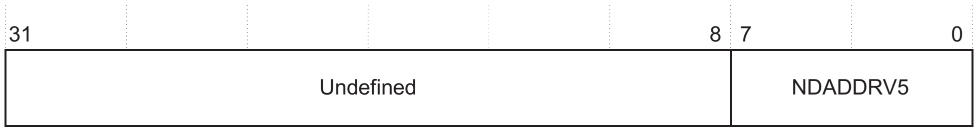


Figure 3-29 MPMCNANDAddress2 Register bit assignments

Table 3-32 lists the register bit assignments.

Table 3-32 MPMCNANDAddress2 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined. Write as zero.
[7:0]	Fifth address vector (NDADDRV5)	`0x00`-`0xFF` = NAND flash fifth address vector (reset value on **nPOR** is `0x0`). This value is only used if the NDADDRPH bits of the MPMCNANDControl Register are set to five.

3.3.30 NAND Memory Timing Value 1 Register

The 25-bit, read/write, MPMCNANDTiming1 Register enables you to program the first set of NAND flash timing parameters. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. Software must poll the NDTX bit in the MPMCNANDStatus Register before programming the timing register. This register is accessed with zero wait states.

Figure 3-30 shows the register bit assignments.

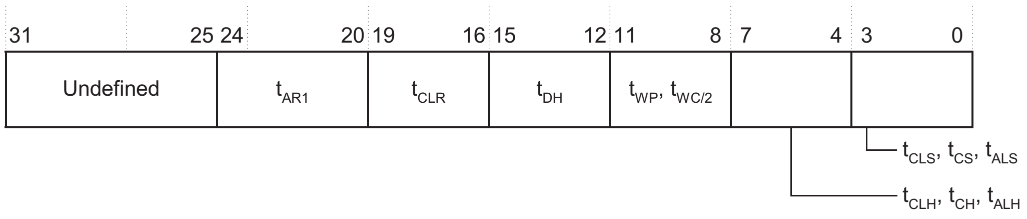


Figure 3-30 MPMCNANDTiming1Register bit assignments

Table 3-33 lists the register bit assignments.

Table 3-33 MPMCNANDTiming1Register bit assignments

Bits	Name	Description
[31:25]	-	Read undefined. Write as zero.
[24:20]	NAND ID read time, ALE to RE falling edge (t_{AR1})	`0x00-0x1E` = n **HCLK** cycles `0x1F` = 31 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameter t_{AR1} on all NAND flash devices in the system must be programmed. This parameter is used when the NDIDRD bit in the MPMCNDControl Register is set, indicating that the current transfer is an ID read.
[19:16]	NAND status read time, CLE LOW to RE LOW (t_{CLR})	`0x0-0xE` = n **HCLK** cycles `0xF` = 15 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameter t_{CLR} on all NAND flash devices in the system must be programmed.[a] This parameter is used when the current transfer is a status read, detected by the memory controller as a transfer without an address phase.
[15:12]	NAND data hold time from WE rising edge (t_{DH})	`0x0-0xE` = n **HCLK** cycles `0xF` = 15 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameter t_{DH} on all NAND flash devices in the system must be programmed.
[11:8]	NAND write enable pulse width time (t_{WP}, $t_{WC/2}$)	`0x0-0xE` = (n+1) **HCLK** cycles `0xF` = 16 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameters t_{WP} and half of t_{WC} on all NAND flash devices in the system must be programmed.
[7:4]	NAND control hold time, CLE, CE, and ALE from WE rising edge (t_{CLH}, t_{CH}, t_{ALH})	`0x0-0xE` = n **HCLK** cycles `0xF` = 15 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameters t_{CLH}, t_{CH}, and t_{ALH} on all NAND flash devices in the system must be programmed.
[3:0]	NAND control setup time, CLE, CE, and ALE to WE falling edge (t_{CLS}, t_{CS}, t_{ALS})	`0x0-0xE` = n **HCLK** cycles `0xF` = 15 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameters t_{CLS}, t_{CS}, and t_{ALS} on all NAND flash devices in the system must be programmed.

a. Some devices use the parameter t_{CLS} instead for status reads, but this value must still be programmed into the t_{CLR} bits even though t_{CLS} is previously defined in bits [3:0] of this register.

3.3.31 NAND Memory Timing Value 2 Register

The 18-bit, read/write, MPMCNANDTiming1 Register enables you to program the second set of NAND flash timing parameters. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the MPMC is idle, and then entering low-power or disabled mode. Software must poll the NDTX bit in the MPMCNANDStatus Register before programming the timing register. This register is accessed with zero wait states.

Figure 3-31 shows the register bit assignments

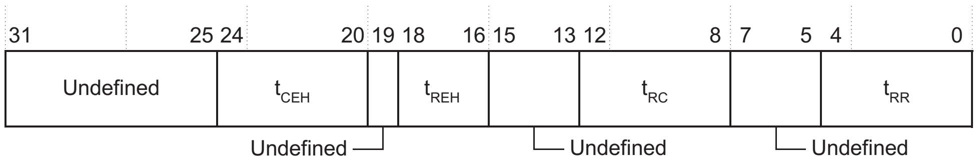


Figure 3-31 MPMCNANDTiming2Register bit assignments

Table 3-34 lists the register bit assignments.

Table 3-34 MPMCNANDTiming2Register bit assignments

Bits	Name	Description
[31:25]	-	Read undefined. Write as zero.
[24:20]	NAND CE HIGH hold time at last serial read (t_{CEH})	`0x00`-`0x1E` = (n+1) **HCLK** cycles`0x1F` = 32 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameter t_{CEH} on all NAND flash devices in the system must be programmed.
[19]	-	Read undefined. Write as zero.
[18:16]	NAND RE HIGH hold time (t_{REH})	`0x0`-`0x6` = (n+1) **HCLK** cycles`0x7` = 8 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameter t_{REH} on all NAND flash devices in the system must be programmed.
[15:13]	-	Read undefined. Write as zero.

Table 3-34 MPMCNANDTiming2Register bit assignments (continued)

Bits	Name	Description
[12:8]	NAND read cycle time (t_{RC})	`0x00-0x1E` = (n+1) **HCLK** cycles`0x1F` = 32 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameter t_{RC} on all NAND flash devices in the system must be programmed.
[7:5]	-	Read undefined. Write as zero.
[4:0]	NAND ready to RE falling edge (t_{RR})	`0x00-0x1E` = n **HCLK** cycles`0x1F` = 31 **HCLK** cycles (reset value on **nPOR**). The worst case value for the parameter t_{RR} on all NAND flash devices in the system must be programmed.

Note

No timing value is specified for the read access time t_{REA}, because this is calculated from the t_{RC} and t_{REH} values. For all read transfers, t_{REA}=t_{RC}-t_{REH}. Some devices specify a longer read access time for ID read transfers. For these devices, an increased t_{RC} value must be programmed for the ID read. It can then be set to the standard read value for all subsequent standard data reads. Some devices might not specify a requirement for the t_{CEH} parameter. The timing value can be programmed to zero for these devices to reduce the read access time.

3.3.32 NAND Status Information Register

The six-bit, read-only, MPMCNANDStatus Register enables you to read various NAND status information. This register is accessed with zero wait states.

Figure 3-32 shows the register bit assignments.

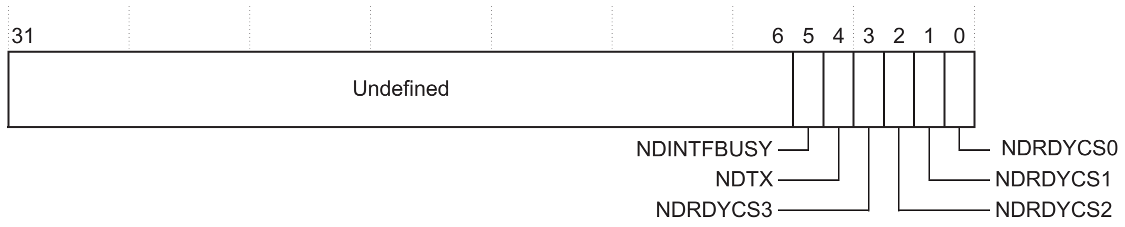


Figure 3-32 MPMCNANDStatus Register bit assignments

Table 3-35 lists the register bit assignments.

Table 3-35 MPMCNANDStatus Register bit assignments

Bits	Name	Description
[31:6]	-	Read undefined.
[5]	NAND interface busy/idle (NDINTFBUSY)	0 = NAND interface is idle (reset value on **nPOR**) 1 = NAND interface is busy. This bit indicates when the NAND interface is busy or idle. It determines when it is safe for the control and address registers to be updated during a NAND access. The interface is typically idle during a device busy phase (waiting for the NAND flash internal operation to complete), or when the device is waiting for a write transfer to be performed. The interface is typically busy when a start command has been issued but the NAND interface has not been granted control of the external bus to initiate the transfer, while command and address phases are being performed, or during a data access. The control and address registers must not be updated during a control or address phase. Updating results in unpredictable behavior.
[4]	NAND transfer (NDTX)	0 = no NAND transfers are performed (reset value on **nPOR**) 1 = a NAND transfer is in progress. This bit indicates when a NAND transfer is in progress, and can be used to determine when it is possible to start a new NAND flash transfer, or cleanly enter the low-power or disabled mode in conjunction with the B bit of the MPMCStatus register.
[3]	Chip select 3 NAND ready status (NDRDYCS3)	0 = NAND device is ready (reset value on **nPOR**) 1 = NAND device is busy. This status bit is an inverted registered version of the chip select 3 device ready/busy output.
[2]	Chip select 2 NAND ready status (NDRDYCS2)	0 = NAND device is ready (reset value on **nPOR**) 1 = NAND device is busy. This status bit is an inverted registered version of the chip select 2 device ready/busy output.
[1]	Chip select 1 NAND ready status (NDRDYCS1)	0 = NAND device is ready (reset value on **nPOR**) 1 = NAND device is busy. This status bit is an inverted registered version of the chip select 1 device ready/busy output.
[0]	Chip select 0 NAND ready status (NDRDYCS0)	0 = NAND device is ready (reset value on **nPOR**) 1 = NAND device is busy. This status bit is an inverted registered version of the chip select 0 device ready/busy output.

3.3.33 AHB Control Registers 0-8

The MPMCAHBControl0-8 Registers are single-bit, read/write registers used to control the AHB interface operation. The register fields can be altered during normal operation.

Figure 3-33 shows the register bit assignments.

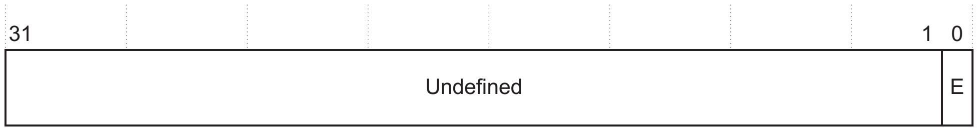


Figure 3-33 MPMCAHBControl0-8 Register bit assignments

Table 3-36 lists the register bit assignments.

Table 3-36 MPMCAHBControl0-8 Register bit assignments

Bits	Name	Description
[31:1]	-	Read undefined. Write as zero.
[0]	Buffer enable (E)	0 = disable buffer (reset value on **nPOR**) 1 = enable buffer and zero wait write states[a]

a. For 32 and 64-bit masters that use separate read and write ports, the associated AHB port buffers must be disabled to ensure coherency with the data read port. For the ARM11 data write port, this bit must be set to 0 to ensure coherency with the data write port.

The HPROT[2] bit is used (bufferable signals) to determine if merging takes place, and the E bit is used to determine if the AHB write buffer is used to hold the transfer. Table 3-37 shows the possible transfer types.

Table 3-37 Transfer types

E bit	HPROT [2]	Transfer
0	0	Nonbuffered, no data merging
0	1	Nonbuffered, data merging allowed
1	0	Buffered, no data merging
1	1	Buffered, data merging allowed

Note

In Table 3-37 on page 3-49:

- Buffered means **HREADY** is returned immediately for first or last transfer.
- Nonbuffered means **HREADY** is held off until transfer is placed in memory.
- Merging allowed means non word burst transfers are converted into word transfers (INCR4 BYTE is passed as 1 word write).
- No data merging means each transfer of the burst is passed through as is (INCR4 BYTE is passed as 4 byte writes).

3.3.34 AHB Status Registers 0-8

The MPMCAHBStatus0-8 Registers are single-bit, read-only registers and provide AHB interface status information.

Figure 3-34 shows the register bit assignments.

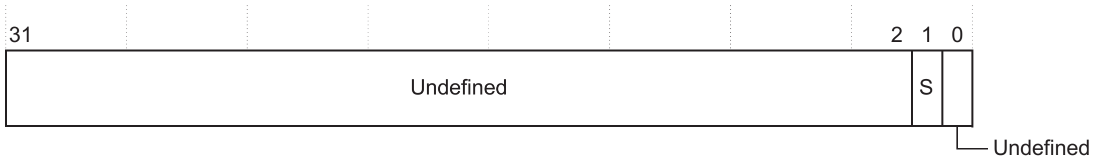


Figure 3-34 MPMCAHBStatus0-8 Register bit assignments

Table 3-38 lists the register bit assignments.

Table 3-38 MPMCAHBStatus0-8 Register bit assignments

Bits	Name	Description
[31:2]	-	Read undefined. Write as zero.
[1]	Buffer status (S)	0 = buffer empty (reset value on **nPOR**) 1 = buffer contains data
[0]	-	Read undefined. Write as zero.

3.3.35 AHB Timeout Registers 0-8

The MPMCAHBTimeOut0-8 Registers are ten-bit, read/write registers that are used to ensure that each AHB port is serviced within a programmed number of cycles. When an AHB request goes active the values in the MPMCAHBTimeOut0-8 Registers are loaded into a down counter. If the transfer is not processed by the time the TimeOut has counted down to 0, the priority of the AHB port is increased until the request is serviced.

These registers therefore enable the transaction latency, and indirectly the bandwidth, for a particular port to be defined. The value programmed into these registers depends on the latency required for the particular port.

The register fields can be altered during normal operation.

Figure 3-35 shows the register bit assignments.

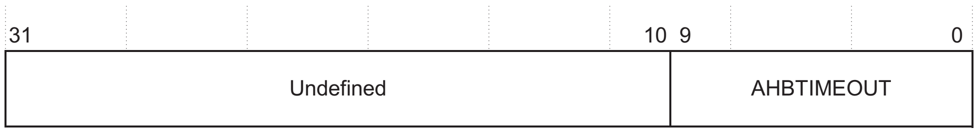


Figure 3-35 MPMCAHBTimeOut0-8 Register bit assignments

Table 3-39 lists the register bit assignments.

Table 3-39 MPMCAHBTimeOut0-8 Register bit assignments

Bits	Name	Description
[31:10]	-	Read undefined. Write as zero.
[9:0]	AHB time out (AHBTIMEOUT)	`0x000` = time out disabled (reset value on **nPOR**) `0x3FF` = 1-1023 **HCLK** cycles before time out is reached

3.3.36 Additional Peripheral Identification Registers

The MPMCPeriphID4-7 Registers are four eight-bit read-only registers, that span address locations `0xFD0`-`0xFDC`. The registers can conceptually be treated as a single register that holds a 32-bit additional peripheral ID value.

Figure 3-36 on page 3-52 shows the register bit assignments.

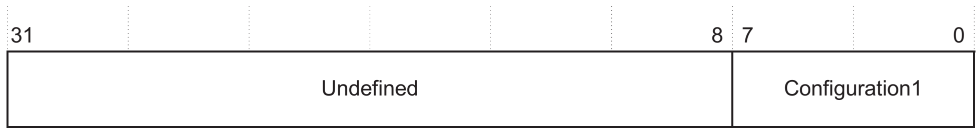


Figure 3-36 Conceptual MPMC Additional Peripheral ID Register bit assignments

Table 3-40 lists the register bit assignments.

Table 3-40 Conceptual MPMC Additional Peripheral ID Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined
[7:0]	Configuration1	Additional peripheral configuration information

The configuration options are peripheral-specific. The four, 8-bit Peripheral Identification Registers are described in the following subsections:

- *Additional Peripheral Identification Register 4*
- *Additional Peripheral Identification Register 5-7* on page 3-53.

Additional Peripheral Identification Register 4

The MPMCPeriphID4 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-37 shows the register bit assignments.

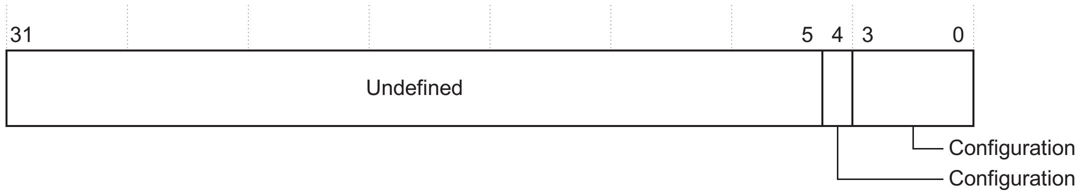


Figure 3-37 MPMCPeriphID4 Register bit assignments

Table 3-41 lists the register bit assignments.

Table 3-41 MPMCPeriphID4 Register bit assignments

Bits	Name	Description
[31:5]	-	Read undefined.
[4]	Configuration	MBX Interface Port: 0 = no 1 = yes For GX176, this field is set to 1.
[3:0]	Configuration	Number of memory ports: 0000-1111 = n + 1 memory ports 1001 = 10 memory ports (value for GX176).

Additional Peripheral Identification Register 5-7

The MPMCPeriphID5-7 Registers are reserved. Table 3-42 shows the bit assignments for the MPMCPeriphID5-7 Registers.

Table 3-42 MPMCPeriphID5-7 Register bit assignments

Bits	Name	Description
[31:0]	-	Read undefined.

3.3.37 Peripheral Identification Registers

The MPMCPeriphID0-3 Registers are four eight-bit read-only registers, that span address locations `0xFE0-0xFEC`. The registers can conceptually be treated as a single register that holds a 32-bit peripheral ID value.

Table 3-43 shows the bit assignments for the conceptual 32-bit MPMC Peripheral Identification Register.

Table 3-43 Conceptual MPMC Peripheral ID Register bit assignments

Bits	Name	Description
[31:24]	Configuration	Configuration options are peripheral-specific. See *Peripheral Identification Register 3* on page 3-56.
[23:20]	Revision	The peripheral revision number is revision dependent.
[19:12]	Designer	Designer ID number. This is `0x41` for ARM.
[11:0]	Part number	Identifies the peripheral. The part number for GX176 is `0x176`.

Figure 3-38 shows the correspondence between bits of the MPMCPeriphID0-3 Registers and the conceptual 32-bit MPMC Peripheral ID Register.

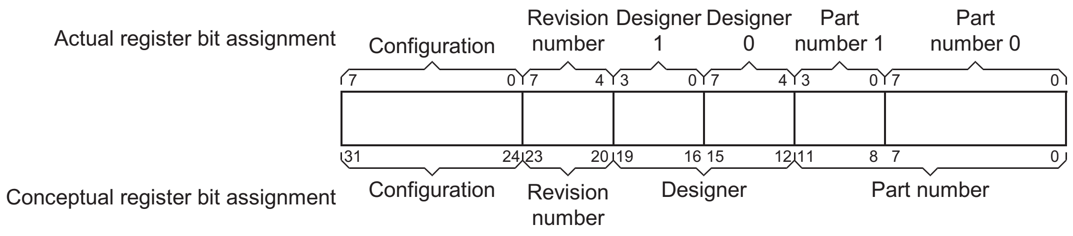


Figure 3-38 Peripheral identification register bit assignment

The four 8-bit Peripheral Identification Registers are described in the following subsections:

- *Peripheral Identification Register 0*
- *Peripheral Identification Register 1* on page 3-55
- *Peripheral Identification Register 2* on page 3-55
- *Peripheral Identification Register 3* on page 3-56.

Peripheral Identification Register 0

The MPMCPeriphID0 Register is hard-coded and the fields in the register determine the reset value. Table 3-44 shows the bit assignments for the MPMCPeriphID0 Register.

Table 3-44 MPMCPeriphID0 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined
[7:0]	PartNumber0	These bits read back as `0x76`

Peripheral Identification Register 1

The MPMCPeriphID1 Register is hard-coded and the fields in the register determine the reset value. Table 3-45 shows the bit assignments for the MPMCPeriphID1 Register.

Table 3-45 MPMCPeriphID1 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined
[7:4]	Designer0	These bits read back as `0x1`
[3:0]	PartNumber1	These bits read back as `0x1`

Peripheral Identification Register 2

The MPMCPeriphID2 Register is hard-coded and the fields in the register determine the reset value. Table 3-46 shows the bit assignments for the MPMCPeriphID2 Register.

Table 3-46 MPMCPeriphID2 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined
[7:4]	Revision	These bits read back as the revision number, that can be 0-15
[3:0]	Designer1	These bits read back as `0x4`

Peripheral Identification Register 3

The MPMCPeriphID3 Register is hard-coded and the fields in the register determine the reset value. Table 3-47 shows the bit assignments for the MPMCPeriphID3 Register.

Table 3-47 MPMCPeriphID3 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined.
[7]	Configuration	Static memory controller: 0 = no 1 = yes. For GX176 this field is set to 1.
[6:4]	Configuration	Indicates the AHB master bus width: 000 = 32-bit wide 001 = 64-bit wide 010 = 128-bit wide 011 = 256-bit wide 100 = 512-bit wide 101 = 1024-bit wide 110-111 = reserved. For GX176 this field is set to 001.
[3:0]	Configuration	Number of AHB slave ports: 0000 = 1 slave port 0001 = 2 slave ports 0010 = 3 slave ports 0011 = 4 slave ports 0100 = 5 slave ports 0101 = 6 slave ports 0110 = 7 slave ports 0111 = 8 slave ports 1000 = 9 slave ports 1001 = 10slave ports 1010 = 11 slave ports 1011 = 12 slave ports 1100 = 13 slave ports 1101 = 14 slave ports 1110 = 15 slave ports 1111 = 16 slave ports For GX176 this field is set to 1001.

3.3.38 PrimeCell Identification Registers 0-3

The MPMCPCellID0-3 Registers are four eight-bit wide registers, that span address locations `0xFF0-0xFFC`. The registers can conceptually be treated as a single register that holds a 32-bit PrimeCell ID value. The register can be used for automatic BIOS configuration. The MPMCPCellID Register is set to `0xB105F00D`.

The register can be accessed with one wait state.

Figure 3-39 on page 3-57 shows the register bit assignments.

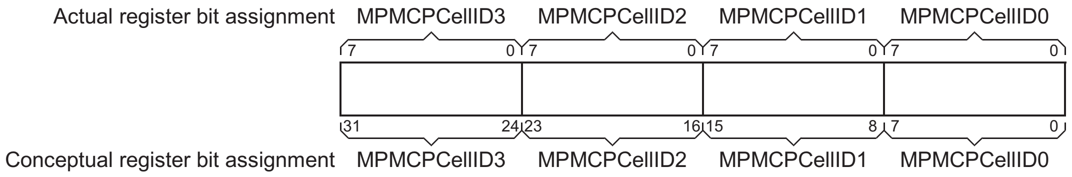


Figure 3-39 Conceptual PrimeCell ID Register bit assignments

Table 3-48 lists the register bit assignments.

Table 3-48 Conceptual PrimeCell ID Register bit assignments

PrimeCell ID Register		MPMCPCellID0-3 Register		
Bits	**Reset value**	**Register**	**Bits**	**Description**
-	-	MPMCPCellID3	[31:8]	Read undefined
[31:24]	`0xB1`	MPMCPCellID3	[7:0]	These bits read back as `0xB1`
-	-	MPMCPCellID2	[31:8]	Read undefined
[23:16]	`0x05`	MPMCPCellID2	[7:0]	These bits read back as `0x05`
-	-	MPMCPCellID1	[31:8]	Read undefined
[15:8]	`0xF0`	MPMCPCellID1	[7:0]	These bits read back as `0xF0`
-	-	MPMCPCellID0	[31:8]	Read undefined
[7:0]	`0x0D`	MPMCPCellID0	[7:0]	These bits read back as `0x0D`

Chapter 4
Programmer's Model for Test

This chapter describes the additional logic for integration testing. It contains the following sections:

4.1 MPMC test harness overview

The additional logic for functional verification and integration vectors enables:

- capture of input signals to the block
- stimulation of the output signals.

The integration vectors provide a way of verifying that the MPMC is correctly wired into a system. This is done by separately testing three groups of signals:

AMBA signals

These are the AMBA bus signals.

Non-AMBA intra-chip signals

Non-AMBA intra-chip signals are on-chip signals that are not part of the AMBA bus, for example, the interrupt request signals.

Primary inputs and outputs The primary input and output signals are those that go off and on the chip.

4.1.1 AMBA test strategy

These signals are tested by performing various AMBA bus transactions.

4.1.2 Non-AMBA intra-chip integration test strategy

Non-AMBA intra-chip signals are on-chip signals that are not part of the AMBA bus, for example, the interrupt request signals.

Many of these signals are difficult to control and observe. Therefore PrimeCells have integration test logic to make this task easier. These test features are enabled by programming the MPMC Integration Test Control Register (MPMCITCR). The intra-chip input signal values can then be read using the MPMCITIP1 register. The intra-chip output signals can be controlled using the MPMCITOP register.

4.1.3 Primary I/O test strategy

The primary I/O signals are tested by performing a sequence of memory accesses.

4.2 Production test

The MPMC is designed to simplify:

- insertion of scan test cells
- use of *Automatic Test Pattern Generation* (ATPG).

Scan insertion and ATPG is the recommended method of manufacturing test.

4.3 Test registers

The MPMC test registers are memory-mapped as shown in Table 4-1.

Table 4-1 Test registers memory map

Name	Offset from base	Type	Reset value	Description
MPMCITCR	`0xF00`	Read/write	`0x0`	See *Test Control Register*
MPMCITIP1	`0xF20`	Read/write	`0x00000000`	See *Test Input 1 Register* on page 4-5
MPMCITIP2	`0xF24`	Read/write	`0x0000`	See *Test Input 2 Register* on page 4-8
MPMCITOP	`0xF40`	Read/write	`0x000`	See *Test Output Register* on page 4-10
MPMCITScratch	`0xF60`	Read/write	`0x00000000`	See *Test Scratch Register* on page 4-12

4.3.1 Test Control Register

The MPMCITCR Register is a single-bit read/write control register. The T bit in this register controls the input test multiplexors. This register must only be used in test mode. The register can be accessed with one wait state.

Figure 4-1 shows the register bit assignments.

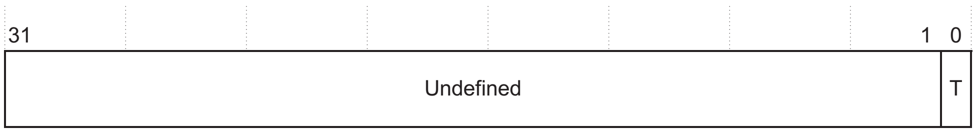


Figure 4-1 MPMCITCR Register bit assignments

Table 4-2 lists the register bit assignments.

Table 4-2 MPMCITCR Register bit assignments

Bits	Name	Description
[31:1]	-	Read undefined. Write as zero.
[0]	Test control register (T)	0 = normal mode (reset value on **nPOR** or **HRESETn**) 1 = test mode

4.3.2 Test Input 1 Register

The MPMCITIP1 Register is a 32-bit read/write register. This register must only be used in test mode. The register can be accessed with one wait state.

Figure 4-2 shows the register bit assignments.

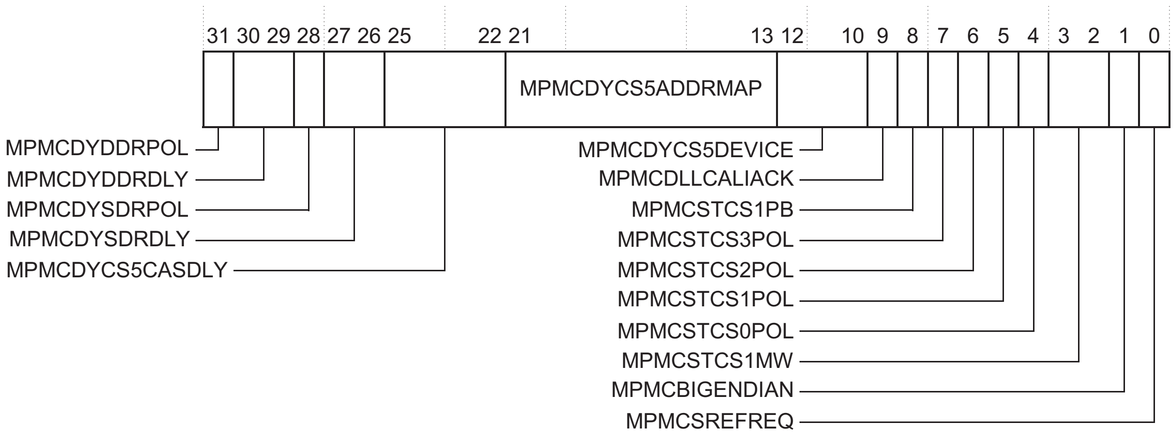


Figure 4-2 MPMCITIP1 Register bit assignments

Table 4-3 lists the register bit assignments.

Table 4-3 MPMCITIP1 Register bit assignments

Bits	Name	Description
[31]	MPMCDYDDRPOL	DDR-SDRAM pad interface polarity signal **MPMCDYDDRPOL**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYDDRPOL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[30:29]	MPMCDYDDRDLY	DDR-SDRAM pad interface configuration signal **MPMCDYDDRDLY**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYDDRDLY** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[28]	MPMCDYSDRPOL	SDR-SDRAM pad interface polarity signal **MPMCDYSDRPOL**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYSDRPOL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[27:26]	MPMCDYSDRDLY	SDR-SDRAM pad interface configuration signal **MPMCDYSDRDLY**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYSDRDLY** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[25:22]	MPMCDYCS5CASDLY	Chip select5 CAS delay signal **MPMCDYCS5CASDLY**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYCS5CASDLY** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[21:13]	MPMCDYCS5ADDRMAP	Chip select5 address map signal **MPMCDYCS5ADDRMAP**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYCS5ADDRMAP** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[12:10]	MPMCDYCS5DEVICE	Chip select5 device signal **MPMCDYCS5DEVICE**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDYCS5DEVICE** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.

Table 4-3 MPMCITIP1 Register bit assignments (continued)

Bits	Name	Description
[9]	MPMCDLLCALIACK	Value of DLL calibration acknowledge signal **MPMCDLLCALIBACK**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDLLCALIBACK** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[8]	MPMCSTCS1PB	Polarity of byte lane for chip select1 signal **MPMCSTCS1PB**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS1PB** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[7]	MPMCSTCS3POL	Polarity of chip select3 signal **MPMCSTCS3POL**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS3POL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[6]	MPMCSTCS2POL	Polarity of chip select2 signal **MPMCSTCS2POL**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS2POL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH
[5]	MPMCSTCS1POL	Polarity of chip select1 signal **MPMCSTCS1POL**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS1POL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH
[4]	MPMCSTCS0POL	Polarity of chip select0 signal **MPMCSTCS0POL**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS0POL** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.

Table 4-3 MPMCITIP1 Register bit assignments (continued)

Bits	Name	Description
[3:2]	MPMCSTCS1MW	Chip select one memory width **MPMCSTCS1MW**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSTCS1MW** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[1]	MPMCBIGENDIAN	Big-endian enable signal **MPMCBIGENDIAN**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCBIGENDIAN** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH
[0]	MPMCSREFREQ (SR)	Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSREFREQ** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.

4.3.3 Test Input 2 Register

The MPMCITIP2 Register is a 15-bit read/write register. This register must only be used in test mode. The register can be accessed with one wait state.

Figure 4-3 shows the register bit assignments.

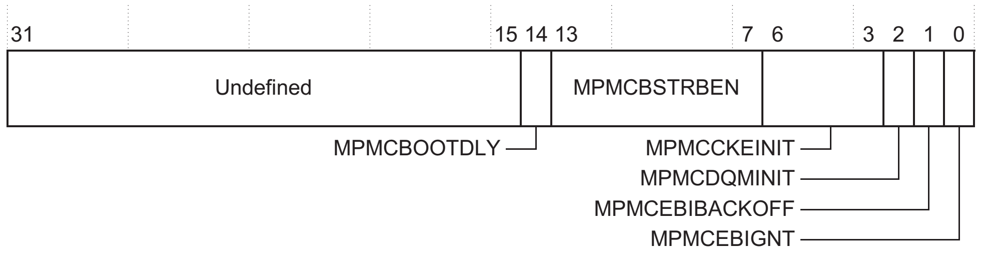


Figure 4-3 MPMCITIP2 Register bit assignments

Table 4-4 lists the register bit assignments.

Table 4-4 MPMCITIP2 Register bit assignments

Bits	Name	Description
[31:15]	-	Read undefined. Write as zero.
[14]	MPMCBOOTDLY	Boot delay signal **MPMCBOOTDLY**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCBOOTDLY** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[13:7]	MPMCBSTRBEN [6:0]	Byte Strobe Enable Signal **MPMCBSTRBENx**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCBSTRBEN[6:0]** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[6:3]	MPMCCKEINIT	Clock enable signal **MPMCCKEINIT**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCCKEINIT** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.

Table 4-4 MPMCITIP2 Register bit assignments (continued)

Bits	Name	Description
[2]	MPMCDQMINIT	Data mask signal **MPMCDQMINIT**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDQMINIT** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH 1 = Set this field if the MPMCITCR T bit is HIGH.
[1]	MPMCEBIBACKOFF	EBI backoff signal **MPMCEBIBACKOFF**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCEBIBACKOFF** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH 1 = Set this field if the MPMCITCR T bit is HIGH.
[0]	MPMCEBIGNT	EBI grant signal **MPMCEBIGNT**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCEBIGNT** input signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH 1 = Set this field if the MPMCITCR T bit is HIGH.

4.3.4 Test Output Register

The MPMCITOP Register is a four-bit read/write register. This register must only be used in test mode. The register can be accessed with one wait state.

Figure 4-4 on page 4-11 shows the register bit assignments.

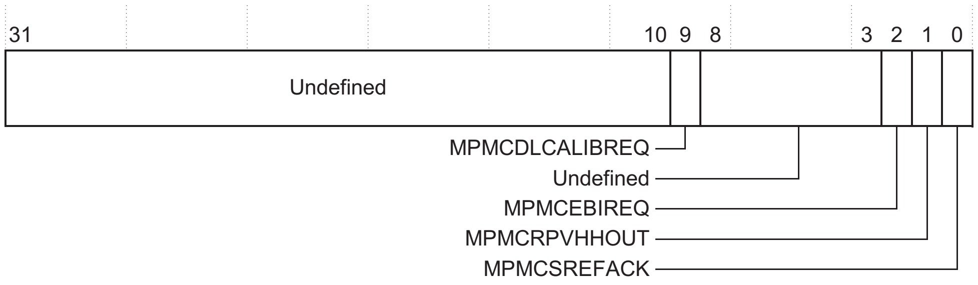


Figure 4-4 MPMCITOP Register bit assignments

Table 4-5 lists the register bit assignments.

Table 4-5 MPMCITOP Register bit assignments

Bits	Name	Description
[31:10]	-	Read undefined. Write as zero.
[9]	MPMCDLLCALIBREQ	Value of DLL calibration request signal **MPMCDLLCALIBREQ**. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCDLLCALIBREQ** output signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field if the MPMCITCR T bit is HIGH. 1 = Set this field if the MPMCITCR T bit is HIGH.
[8:3]	-	Read undefined. Write as zero.

Table 4-5 MPMCITOP Register bit assignments (continued)

Bits	Name	Description
[2]	MPMCEBIREQ	Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCEBIREQ** output signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field and the **MPMCEBIREQ** output signal if the MPMCITR T bit is HIGH 1 = Set this field and the **MPMCEBIREQ** output signal if the MPMCITR T bit is HIGH.
[1]	MPMCRPVHHOUT	Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCRPVHHOUT** output signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field and the **MPMCRPVHHOUT** output signal if the MPMCITR T bit is HIGH. 1 = Set this field and the **MPMCRPVHHOUT** output signal if the MPMCITR T bit is HIGH.
[0]	MPMCSREFACK (SA)	Self-refresh acknowledge. Read: Read the value of this field if the MPMCITCR T bit is HIGH. Read the value of the **MPMCSREFACK** output signal if the MPMCITCR T bit is LOW. Write: 0 = Clear this field and the **MPMCSREFACK** output signal if the MPMCITR T bit is HIGH. 1 = Set this field and the **MPMCSREFACK** output signal if the MPMCITR T bit is HIGH.

4.3.5 Test Scratch Register

The MPMCITScratch Register is a 32-bit general-purpose read/write register. This register is accessed with one wait state.

Table 4-6 lists the bit assignments for the MPMCITScratch Register.

Table 4-6 MPMCITScratch Register bit assignments

Bits	Name	Description
[31:0]	Test Scratch Register	Test scratch register: `0x00000000` = Reset value on **nPOR** or **HRESETn** `0x00000000-0xFFFFFFFF` = All bits are general-purpose read/write. This register is used to store and retrieve data during device test.

Appendix A
Signal Descriptions

This appendix describes the signals that interface with the ARM MPMC block. It contains the following sections:

A.1 AHB register signals

Table A-1 describes the AHB register signals.

Table A-1 AHB register signal descriptions

Name	Type	Source/ destination	Description
HADDRREG[11:2]	Input	AHB master layer	The AHB address bus.
HRDATAREG[31:0]	Output	AHB master layer	Read databus. The read databus is used to transfer data from the MPMC to the bus master during read operations.
HREADYINREG	Input	AHB slave layer	Transfer done. When HIGH, the **HREADYIN** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer. An alternate slave generates this signal.
HREADYOUTREG	Output	AHB master and AHB master layer	Transfer done. When HIGH, the **HREADYOUT** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESPREG[1:0]	Output	AHB master layer	Transfer response. The transfer response provides additional information on the status of a transfer. The SDRAM controller can respond with either the OKAY or ERROR responses. The ERROR response is generated when the transfer size is not 32 bits wide. **HRESPREG[1]** = 0, because SPLIT and RETRY responses are not supported.
HSELMPMCREG	Input	AHB decoder	Slave select. MPMC register select signal. This signal indicates an access to the memory controllers control registers.
HSIZEREG[2:0]	Input	AHB master layer	Transfer size. Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). Only word (32-bit) transfers are allowed (**HSIZE[2:0]** = `0b010`) to access the MPMC registers. Transfer sizes other than 32 bits generate an ERROR response on **HRESPREG[0]**.
HTRANSREG	Input	AHB master layer	Transfer type. Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY. Only **HTRANS[1]** is used, indicating if a transfer is required or not.
HWDATAREG[31:0]	Input	AHB master layer	Write databus. The write databus is used to transfer data from the master to the bus slaves during write operations.
HWRITEREG	Input	AHB master layer	Transfer direction. When HIGH this signal indicates a write transfer and when LOW a read transfer.

A.2 AHB memory signals

Table A-2 describes the AHB memory signals. The signal names for each port can be found by substituting the port number, 0, 1, 2, through to 8 for the symbol x. For signals specific to 64-bit ports, the symbol x can only have a value of 3 to 6. Unused ports are disabled by connecting their inputs to logic 0. The MPMC does not support RETRY or SPLIT transactions.

Table A-2 AHB memory signal descriptions

Name	Type	Source/ destination	Description
HADDRx[28:0]	Input	AHB master layer	The AHB address bus.
HBSTRBx[7:0]	Input	ARM11 AHB master	Byte lane enables for individual byte lanes [63:56], [55:48], [47:40], [39:32], [31:24], [23:16], [15:8], and [7:0].
HBURSTx[2:0]	Input	AHB master layer	Burst type. Indicates if the transfer forms part of a burst. 4, 8, and 16 beat bursts are supported and the burst can be either incrementing or wrapping. Burst type INCR, incrementing burst of unspecified length, is interpreted as INCR4 by the memory controller.
HDOMAINx[1:0]	Input	Arbiter	The **HDOMAINx[1:0]** signals from the system indicate which bus domain is currently performing a transfer and determine which ARM11 compliant master is executing an Exclusive Access transfer. Support for monitoring a maximum of four domains is provided.
HMASTLOCKx	Input	AHB master layer	Locked transfers. When HIGH this signal indicates that the master requires locked access to the SDRAM and no other master must be granted the bus until this signal is LOW.
HPROTx[5] **HPROTx[3:2]**	Input	AHB master	Protection control signals. **HPROT[3:2]** provide additional information about a bus access. These signals indicate whether the transfer is: • an opcode fetch or a data access • a privileged mode access or a User mode access • a cacheable access or a noncacheable access • a bufferable access or a nonbufferable access. For ARM11 compliant bus masters, **HPROT[5]** indicates an outer memory load/store exclusive transfer.

Table A-2 AHB memory signal descriptions (continued)

Name	Type	Source/ destination	Description
HRDATAx[63:0]	Output	AHB master layer	Read databus. The read databus is used to transfer data from the MPMC to the bus master during read operations. 32-bit ports use the lower 32 bits of the 64-bit databus, **HRDATA[31:0]**
HREADYINx	Input	AHB slave layer	Transfer done. When HIGH, the **HREADYIN** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer. An alternate slave generates this signal.
HREADYOUTx	Output	AHB master layer	Transfer done. When HIGH, the **HREADYOUT** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESPx[1:0][a]	Output	AHB master layer	Transfer response. The transfer response provides additional information on the status of a transfer. The SDRAM controller can respond with either the OKAY or ERROR responses. The ERROR response is generated when: • the transfer size is greater than allowed • the memory access is a write to a write-protected region • the memory is accessed with the MCEnable bit disabled or the LowPowerMode bit is asserted.
HRESPx[2]	Output	AHB master layer	Transfer response. The transfer response provides additional information on the status of an exclusive transfer. The STREX response is generated when the port acknowledges an ARM11 store exclusive instruction.
HSELMPMCxCS[7:0]	Input	AHB decoder	Slave select. MPMC select signal. These signals indicate an access to a particular memory bank.
HSELMPMCxG	Input	AHB decoder	Slave select. MPMC global select signal. This signal indicates an access to memory.
HSIZEx[2:0]	Input	AHB master layer	Transfer size. Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit), word (32-bit), or Dword (64-bit). Byte (8-bit), halfword (16-bit), word (32-bit), and Dword (64-bit) transfers are allowed to access the external memory. Transfer sizes greater than 64 bits generate an ERROR response.
HTRANSx[1:0]	Input	AHB master layer	Transfer type. Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY.

Table A-2 AHB memory signal descriptions (continued)

Name	Type	Source/ destination	Description
HUNALIGNx	Input	ARM11 AHB master	Indicates that the current transfer is part of an unaligned data transfer. Unaligned write transfers are not supported by all memory devices.
HWDATAx[63:0]	Input	AHB master layer	Write databus. The write databus is used to transfer data from the master to the bus slaves during write operations. 32-bit ports use the lower 32 bits of the 64-bit databus, **HWDATA[31:0]**.
HWRITEx	Input	AHB master layer	Transfer direction. When HIGH this signal indicates a write transfer and when LOW a read transfer.

a. **HRESPx[1]** = 0 because SPLIT and RETRY responses are not supported.

A.3 MBX Interface Port signals

The MPMC MBX Interface Port interface is a slave port that connects directly to the MBX 3D Graphics Core master port. Table A-3 shows the MBX Interface Port signals.

Table A-3 MBX Interface Port signals

Name	Type	Source/ destination	Description
GADDR[28:3]	Input	MBX 3D Graphics Core	This is the MBX 3D Graphics Core memory interface address bus for reads and writes. It is 29 bits wide, and is a 64-bit aligned (doubleword) address. The value on this bus is captured by the MPMC when **GTRANS** = 1 and **GAREADY** = 1. If the system does not contain an MMU then the top seven bits are not used, and are tied LOW.
GAREADY	Output	MBX 3D Graphics Core	This signal is driven HIGH when the MPMC is ready to receive another transfer from the MBX 3D Graphics Core memory interface controller.
GBSTRB[7:0]	Input	MBX 3D Graphics Core	This is a eight-bit signal which is a byte mask. A HIGH on the relevant byte mask indicates that the byte is valid. There are certain requestors in the MBX 3D Graphics Core that do read-modify-writes and they have a byte granularity. That is, only one or two bytes are valid and therefore the data in the remaining bytes of a 64-bit memory location must be retained. These bits relate directly to the **DQM** signals found on SDRAM devices. The bits of **GBSTRB** relate to the bytes of **GWDATA** as follows: 0 = [7:0] 1 = [15:8] 2 = [23:16] 3 = [31:24]. 4 = [39:32] 5 = [47:40] 6 = [55:48] 7 = [63:56].
GDREADY	Output	MBX 3D Graphics Core	This signal is driven HIGH by the MPMC to indicate that the returned data is valid.
GRDATA[63:0]	Output	MBX 3D Graphics Core	The read data bus is used to transfer data from the MPMC to the MBX 3D Graphics Core during read operations. The value on this bus is captured by the MBX 3D Graphics Core when **GDREADY** = 1.
GSELCS[7:4]	Input	GXI Decoder	This is the MPMC slave select signal. These signals indicate an access to a particular dynamic memory chip select.

Table A-3 MBX Interface Port signals (continued)

Name	Type	Source/ destination	Description
GSELG	Input	GXI Decoder	This is the MPMC global slave select signal. This signal indicates an access to memory.
GTRANS	Input	MBX 3D Graphics Core	This signal, when HIGH, indicates that the current transfer is valid and that the control signals can be sampled by the slave.
GWDATA[63:0]	Input	MBX 3D Graphics Core	The write data bus is used to transfer data from the MBX 3D Graphics Core to the MPMC during write operations. The value on this bus is captured by the MPMC when **GTRANS** = 1 and **GAREADY** = 1.
GWRITE	Input	MBX 3D Graphics Core	This signal defines whether the current transfer is a read or a write. When LOW it signifies a read. When HIGH it signifies a write. The value of this signal is captured by the MPMC when **GTRANS** = 1 and **GAREADY** = 1.

A.4 Miscellaneous signals

The MPMC miscellaneous signals are described in:

- *Tie-off signals*
- *Test signals* on page A-13
- *Clock and reset signals* on page A-13
- *DLL and self-refresh signals* on page A-14
- *External Bus Interface signals* on page A-14.

A.4.1 Tie-off signals

Table A-4 describes the tie-off signals.

Table A-4 Tie-off signal descriptions

Name	Type	Source/ destination	Description
MPMCBIGENDIAN	Input	Reset controller	Endian select. Determines the default endianness of the AHB slave ports and external memory banks. If this bit is HIGH, big-endian mode is selected. The value can be overridden by writing to the N field of the MPMCConfig Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCBOOTDLY	Input	Reset controller	When HIGH, indicates that no accesses must be performed. Used for dynamic memory devices that cannot be accessed immediately after reset. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCBSTRBENx	Input	Reset controller	Byte strobe enable. Enables ARM11 AHB byte strobe inputs **HBSTRB[7:0]** for unaligned data support. Enabling the byte strobes for a given port also fixes that port into byte-invariant mode for ARM11 mixed-endian support.
MPMCCKEINIT[3:0]	Input	Reset controller	Controls the value driven on the **MPMCCKEOUT** signals during a power-up sequence. See the data sheet for the SDRAM memory device being used for the appropriate value.

Table A-4 Tie-off signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCDQMINIT	Input	Reset controller	Defines the power-on level for the **MPMCDQMOUT** signals after **nPOR**. When SDR/DDR memory is in self-refresh mode this value must be 0. For uninitialized SDR/DDR memory this value must be 1.
MPMCDYCS5ADRMAP[8:0]	Input	Reset controller	Indicates the address map for chip select 5. Used for dynamic memory devices. The value can be overridden by writing to the AM field of the MPMCDynamicConfig1 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Configuration Registers 0-3* on page 3-27.
MPMCDYCS5CASDLY[3:0]	Input	Reset controller	Indicates the upper four bits of CAS delay for chip select 5. Used for dynamic memory devices. The value can be overridden by writing to the upper three bits of the CAS field of the MPMCDynamicRasCas1 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory RAS and CAS Delay Registers 0-3* on page 3-32.
MPMCDYCS5DEVICE[2:0]	Input	Reset controller	Indicates the type of device connected to chip select 5. Used for dynamic memory devices. The value can be overridden by writing to the MD field of the MPMCDynamicConfig1 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Configuration Registers 0-3* on page 3-27.

Table A-4 Tie-off signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCDYDDRDLY[1:0][a]	Input	Reset controller	DDR clocking scheme for data capture. Used for dynamic memory devices. The value can be overridden by writing to the DRD field of the MPMCDynamicReadConfig Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Read Configuration Register* on page 3-17.
MPMCDYDDRPOL[a]	Input	Reset controller	The polarity of the register in which the DDR read data is first captured. Used for dynamic memory devices. The value can be overridden by writing to the DRP field of the MPMCDynamicReadConfig Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Read Configuration Register* on page 3-17.
MPMCDYSDRDLY[1:0]	Input	Reset controller	SDR clocking scheme for data capture. Used for dynamic memory devices. The value can be overridden by writing to the SRD field of the MPMCDynamicReadConfig Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Read Configuration Register* on page 3-17.
MPMCDYSDRPOL	Input	Reset controller	The polarity of the register in which the SDR read data is first captured. Used for dynamic memory devices. The value can be overridden by writing to the SRP field of the MPMCDynamicReadConfig Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value. For more information on the tie-off values see *Dynamic Memory Read Configuration Register* on page 3-17.

Table A-4 Tie-off signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCSTCS0POL	Input	Reset controller	Chip select 0 polarity select on **nPOR**. A HIGH value indicates CS active HIGH, a LOW value indicates CS active LOW. Used for static memory devices. The value can be overridden by writing to the PC field of the MPMCStaticConfig0 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCSTCS1MW[1:0]	Input	Reset controller	Chip select 1 memory width selection: 00 = eight-bit wide memory 01 = 16-bit wide memory 10 = 32-bit wide memory 11 = reserved. Used for static memory devices. The value can be overridden by writing to the MW field of the MPMCStaticConfig1 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCSTCS1PB	Input	Reset controller	Chip select 1 byte lane polarity select.0 = for reads all the bits in **nMPMCBLSOUT[3:0]** are HIGH. For writes the respective active bits in **nMPMCBLSOUT[3:0]** are LOW.1 = for reads the respective active bits in **nMPMCBLSOUT[3:0]** are LOW. For writes the respective active bits in **nMPMCBLSOUT[3:0]** are LOW.Used for static memory devices. The value can be overridden by writing to the PB field of the MPMCStaticConfig1 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value.

Table A-4 Tie-off signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCSTCS1POL	Input	Reset controller	Chip select 1 polarity select on power-on reset (**nPOR**). A HIGH value indicates CS active HIGH, a LOW value indicates CS active LOW. Used for static memory devices. The value can be overridden by writing to the PC field of the MPMCStaticConfig1 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCSTCS2POL	Input	Reset controller	Chip select 2 polarity select on **nPOR**. A HIGH value indicates CS active HIGH, a LOW value indicates CS active LOW. Used for static memory devices. The value can be overridden by writing to the PC field of the MPMCStaticConfig2 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value.
MPMCSTCS3POL	Input	Reset controller	Chip select 3 polarity select on **nPOR**. A HIGH value indicates CS active HIGH, a LOW value indicates CS active LOW. Used for static memory devices. The value can be overridden by writing to the PC field of the MPMCStaticConfig3 Register. This signal must not be altered during normal operation. This signal can be generated by registering an input signal on **nPOR** in the reset controller. Alternatively, this signal can be tied to the appropriate value.

a. This signal is reserved for future use and must be tied LOW.

A.4.2 Test signals

Table A-5 describes the test signals.

Table A-5 Test signal descriptions

Name	Type	Source/ destination	Description
MPMCTESTIN	Input	Pad	Test mode, used to place the MPMC into TIC test mode.
MPMCTESTREQA	Input	Pad	Test bus request A. This pin is used in TIC test mode. In test mode this pin is the test bus request A input signal and is required as a dedicated device pin. During normal system operation the **MPMCTESTREQA** signal is used to request entry into the test mode. During test **MPMCTESTREQA** is used, in combination with **MPMCTESTREQB**, to indicate the type of test vector that is applied in the following cycle.
MPMCTESTREQB	Input	Pad	Test bus request B. This pin is used in TIC test mode. During test this signal is used, in combination with **MPMCTESTREQA**, to indicate the type of test vector that is applied in the following cycle. The test bus acknowledge signal gives external indication that the test bus has been granted and also indicates when a test access has completed. When **MPMCTESTACK** is LOW, the current test vector must be extended until **MPMCTESTACK** becomes HIGH.

A.4.3 Clock and reset signals

Table A-6 describes the clock and reset signals.

Table A-6 Clock and reset signal descriptions

Name	Type	Source/ destination	Description
HCLK	Input	Clock source	Bus clock. This clock times all bus transfers. All signal timings are related to the rising edge of **HCLK**.
nHCLK	Input	Clock source	Inverted **HCLK** signal that can be used to capture the read data from SDRAM.
HCLKX2	Input	Clock source	Double frequency clock that is synchronous to **HCLK**. This clock is used for pad interface DDR timings. This clock frequency is **HCLK** x 2 and is synchronous to **HCLK**. It is not required in non-DDR memory systems.

Table A-6 Clock and reset signal descriptions (continued)

Name	Type	Source/ destination	Description
nHCLKX2	Input	Clock source	Inverted double frequency clock that is synchronous to **HCLK**. This clock is used for the negative edge triggered flip-flops in DDR operations. It is not required in non-DDR memory systems.
MPMCHCLK DELAY	Input	Clock source	Delayed version of **HCLK** used in command delayed mode to ensure that SDRAM setup and hold requirements are met.
HRESETn	Input	Reset controller	Reset. The bus reset signal is active LOW and is used to reset the system and the bus.
nPOR	Input	Reset controller	Power-on reset. Active LOW.

A.4.4 DLL and self-refresh signals

Table A-7 describes the DLL and self-refresh signals.

Table A-7 DLL and self-refresh signal descriptions

Name	Type	Source/ destination	Description
MPMCDLLCALIBREQ	Output	DLL block	DLL calibration request.
MPMCDLLCALIBACK	Input	DLL block	DLL calibration acknowledge.
MPMCSREFREQ	Input	Power management unit	Self-refresh request.
MPMCSREFACK	Output	Power management unit	Self-refresh acknowledgement.

A.4.5 External Bus Interface signals

Table A-8 on page A-15 describes the *External Bus Interface* (EBI) signals.

Table A-8 EBI signal descriptions

Name	Type	Source/ destination	Description
MPMCEBIGNT	Input	EBI	The EBI grant signal goes active HIGH when the EBI grants the external memory bus.
MPMCEBIBACKOFF	Input	EBI	The EBI backoff signal goes active HIGH when the EBI wants to remove the memory controller from the memory bus so that another memory controller can be granted the memory bus.
MPMCEBIREQ	Output	EBI	The EBI request signal goes active HIGH when the memory controller requires the memory bus.

A.5 Pad interface and control signals

Table A-9 describes the pad interface and control signals.

Table A-9 Pad interface and control signal descriptions

Name	Type	Source/ destination	Description
MPMCADDROUT[27:0]	Output	Pad	Address output. Used for both static and SDRAM devices. 256Mb maximum per static memory bank. SDRAM uses bits [14:0]. Not used by NAND flash. Auto-precharge connections use the **MPMCAPOUT** output instead of bits [8] or [10] of **MPMCADDROUT**.
MPMCAPOUT	Output	Pad	SDRAM auto-precharge address bit. Micron SyncFlash and V-SyncFlash active terminate command control bit.
MPMCCKEOUT[3:0]	Output	Pad	SDRAM clock enables. Used for SDRAM devices.
MPMCCLKOUT[3:0]	Output	Pad	Positive differential clocks. Four clocks provide support for 16-bit minimum width SDRAM devices when using 64-bit wide memory. 8-bit devices are not supported for 64-bit wide memory.
MPMCDATAIN[63:0]	Input	Pad	Read data from memory. Used for the static memory controller, the dynamic memory controller and the TIC:**MPMCDATAIN[63:32]** = Read upper data word for SDRAM.**MPMCDATAIN[31:0]** = Read lower data word for SDRAM, read upper and lower data word for DDR-SDRAM, read data word for static memory and read data word for TIC. **MPMCDATAIN[7:0]** = Read databus for NAND flash devices. 16-bit NAND flash devices also use **MPMCDATAIN[15:8]**.
MPMCDATAOUT[63:0]	Output	Pad	Data output to memory. Used for the static memory controller, the dynamic memory controller and the TIC:**MPMCDATAOUT[63:32]** = Write upper data word for SDRAM.**MPMCDATAOUT[31:0]** = Write lower data word for SDRAM, write upper and lower data word for DDR-SDRAM, write data word for static memory and write data word for TIC.**MPMCDATAOUT[7:0]** = Control, address and write databus for NAND flash devices. 16-bit NAND flash devices also use **MPMCDATAOUT[15:8]** for write data only.

Table A-9 Pad interface and control signal descriptions (continued)

Name	Type	Source/ destination	Description
MPMCDQMOUT[7:0]	Output	Pad	Data mask output to SDRAMs. Used for SDRAM devices:**MPMCDQMOUT[7:4**] = Data mask output, one per byte, for upper word of SDRAM.**MPMCDQMOUT[3:0]** = Data mask output, one per byte, for lower word of SDRAM or upper and lower word of DDR-SDRAM.
MPMCDQSIN[3:0]	Input	Pad	Input data strobes, one per byte, for DDR-SDRAM upper or lower word.
MPMCDQSOUT[3:0]	Output	Pad	Output data strobe, one per byte, for DDR-SDRAM upper or lower word.
MPMCFBCLKIN[7:0]	Input	Pad	Positive differential fed-back clock. Used for SDRAM devices.
MPMCNDALEOUT	Output	Pad	NAND flash address latch enable.
MPMCNDCLEOUT	Output	Pad	NAND flash command latch enable.
MPMCNDREADYIN [3:0]	Input	Pad	NAND flash device ready outputs, for all four possible devices. Deasserted LOW to indicate that the device is busy. Unused inputs must be tied HIGH.
MPMCRPVHHOUT	Output	Pad	Voltage control for Micro Syncflash reset signal (RP).
nMPMCBLSOUT[3:0]	Output	Pad	Byte lane select, active LOW, for static memories. Used for static memory devices.
nMPMCCASOUT	Output	Pad	Column address strobe. Used for SDRAM devices.
nMPMCCLKOUT[3:0]	Output	Pad	Negative differential clocks. Four clocks provide support for 16-bit minimum width SDRAM devices when using 64-bit wide memory. 8-bit devices are not supported for 64-bit wide memory.

Table A-9 Pad interface and control signal descriptions (continued)

Name	Type	Source/ destination	Description
nMPMCDATAOUTEN [7:0]	Output	Pad	**nMPMCDATAOUTEN[7:4]** = Tristate I/O pad enable for the external memory databus **MPMCDATA[63:32]**, active LOW. Enables the external memory databus byte lanes [63:56], [55:48], [47:40], and [39:32] independently, for upper output data word for SDRAM. **nMPMCDATAOUTEN[3:0]** = Tristate I/O pad enable for the external memory databus **MPMCDATA[31:0]**, active LOW. Enables the external memory databus byte lanes [31:24], [23:16], [15:8], and [7:0] independently, for lower output data word for SDRAM, output data word for static memory or TIC, or upper and lower output data word for DDR-SDRAM. The lower 1 or 2 byte lanes are used for NAND flash
nMPMCDQSIN[3:0]	Input	Pad	Half-cycle delayed input data strobes, one per byte, for DDR-SDRAM upper or lower word.
nMPMCDQSOUTEN[3:0]	Output	Pad	Tristate I/O pad enable for the output data strobes, active LOW. Enables the memory device byte lanes [31:24], [23:16], [15:8], and [7:0] for DDR-SDRAM output data strobes independently.
nMPMCDYCSOUT[3:0]	Output	Pad	Active LOW chip selects for SDRAM. CS[7:4].
nMPMCDYWEOUT	Output	Pad	Write enable for dynamic memory.
nMPMCNDREOUT	Output	Pad	Read enable for NAND flash memory.
nMPMCNDWEOUT	Output	Pad	Write enable for NAND flash memory.
nMPMCOEOUT	Output	Pad	Output enable for static memories. Used for static memory devices.
nMPMCRASOUT	Output	Pad	Row address strobe. Used for SDRAM devices.
nMPMCRPOUT	Output	Pad	Reset power down to SyncFlash memory. Used for the dynamic memory controller.
nMPMCSTCSOUT[3:0]	Output	Pad	Default active LOW chip selects for static memory and NAND flash. CS[3:0].
nMPMCSTWEOUT (MPMCTESTACK)	Output	Pad	Write enable for static memory. (Test bus acknowledge signal for TIC test mode.)

A.6 Test Interface Controller (TIC) AHB signals

Table A-10 describes the TIC signals.

Table A-10 TIC signal descriptions

Name	Type	Source/ destination	Description
HADDRTIC[31:0] Address bus	Output	AHB slave layer	The 32-bit AHB address bus.
HBUSREQTIC Bus request	Output	AHB arbiter	A signal from the TIC to the bus arbiter indicates that it requires the bus.
HBURSTTIC[2:0] Burst type	Output	AHB slave layer	Indicates if the transfer forms part of a burst. The TIC always performs incrementing bursts of unspecified length.
HGRANTTIC Bus grant	Input	AHB arbiter	This signal indicates that the TIC is currently the highest priority master. Ownership of the address and control signals changes at the end of a transfer when **HREADYINTIC** is HIGH, so the TIC gains access to the bus when both **HREADYINTIC** and **HGRANTTIC** are HIGH.
HLOCKTIC Locked transfers	Output	AHB arbiter	When HIGH this signal indicates that the TIC requires locked access to the bus. No other master must be granted the bus until this signal is LOW.
HPROT[3:0] Protection control	Output	AHB slave layer	The protection control signals indicate if the transfer is an opcode fetch or data access, as well as if the transfer is a supervisor mode access or a User mode access. These signals also indicate if the current access is cacheable or bufferable.
HRDATATIC[31:0] Read databus	Input	AHB master layer	The read databus is used to transfer data from the AHB slave to the TIC during read operations.
HREADYINTIC Transfer done	Input	AHB slave layer	When HIGH the **HREADYINTIC** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESPTIC[1:0] Transfer response	Input	AHB slave layer	The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY, and SPLIT.
HSIZETIC[2:0] Transfer size	Output	AHB slave layer	Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit), or word (32-bit). The TIC does not support larger transfer sizes.

Table A-10 TIC signal descriptions (continued)

Name	Type	Source/ destination	Description
HTRANSTIC[1:0] Transfer type	Output	AHB slave layer	Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY. The TIC does not use the BUSY transfer type.
HWDATATIC[31:0] Write databus	Output	AHB slave layer	The write databus is used to transfer data from the master to the bus slaves during write operations.
HWRITETIC Transfer direction	Output	AHB slave layer	When HIGH, this signal indicates a write transfer and when LOW a read transfer.

A.7 Scan test signals

Table A-11 describes the scan test signals. It might be possible to reduce the number of scan chains required if synchronous clock domains are balanced during clock tree insertion.

Table A-11 Scan test signal descriptions

Name	Type	Source/ destination	Description
SCANENABLE	Input	Test controller	Scan enable
SCANINHCLK	Input	Test controller	Scan data input for **HCLK** scan chain
SCANOUTHCLK	Output	Test controller	Scan data output for **HCLK** scan chain
nSCANINHCLK	Input	Test controller	Scan data input for **nHCLK** scan chain
nSCANOUTHCLK	Output	Test controller	Scan data output for **nHCLK** scan chain
SCANINHCLKX2	Input	Test controller	Scan data input for **HCLKX2** scan chain
SCANOUTHCLKX2	Output	Test controller	Scan data output for **HCLKX2** scan chain
nSCANINHCLKX2	Input	Test controller	Scan data input for **nHCLKX2** scan chain
nSCANOUTHCLKX2	Output	Test controller	Scan data output for **nHCLKX2** scan chain
SCANINHCLKDELAY	Input	Test controller	Scan in for **MPMCHCLKDELAY** scan chain
SCANOUTHCLKDELAY	Output	Test controller	Scan data output for **MPMCHCLKDELAY** scan chain
SCANINFBCLKIN[7:0]	Input	Test controller	Scan in for **MPMCFBCLKIN[7:0]** scan chains
SCANOUTFBCLKIN[7:0]	Output	Test controller	Scan out for **MPMCFBCLKIN[7:0]** scan chain
SCANINDQSIN[3:0]	Input	Test controller	Scan in for **MPMCDQSIN[3:0]** scan chain
SCANOUTDQSIN{3:0]	Output	Test controller	Scan out for **MPMCDQSIN[3:0]** scan chain
nSCANINDQSIN[3:0]	Input	Test controller	Scan in for **nMPMCDQSIN[3:0]** scan chain
nSCANOUTDQSIN{3:0]	Output	Test controller	Scan out for **nMPMCDQSIN[3:0]** scan chain

Index