AMBA® LPDDR2 Dynamic Memory Controller DMC-342

Revision: r0p0

Technical Reference Manual

ARM®

ARM DDI 0436A (ID103109)

AMBA LPDDR2 Dynamic Memory Controller DMC-342
Technical Reference Manual

Release Information

The *Change history* table lists the changes made to this book.

Change history

Date	Issue	Confidentiality	Change
17 September 2009	A	Non-Confidential	Initial release for r0p0

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Unrestricted Access is an ARM internal classification.

Product Status

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

Web Address

`http://www.arm.com`

Contents
AMBA LPDDR2 Dynamic Memory Controller DMC-342 Technical Reference Manual

List of Tables
AMBA LPDDR2 Dynamic Memory Controller DMC-342 Technical Reference Manual

List of Figures
AMBA LPDDR2 Dynamic Memory Controller DMC-342 Technical Reference Manual

Preface

This preface introduces the *AMBA LPDDR2 Dynamic Memory Controller DMC-342 Technical Reference Manual*. It contains the following sections:

About this book

This is the *Technical Reference Manual* (TRM) for the *AMBA LPDDR2 Dynamic Memory Controller* (LPDDR2 DMC). The LPDDR2 DMC comprises various systems that you can render to support a single type and size of LPDDR2 SDRAM, or LPDDR SDRAM, on the memory interface.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this book, 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 book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) device that uses the LPDDR2 DMC. The LPDDR2 DMC provides an interface between the *Advanced eXtensible Interface* (AXI™) system bus and external, off-chip, memory devices.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the LPDDR2 DMC and its features.

Chapter 2 *Functional Description*

Read this for an overview of the major functional blocks and the operation of the LPDDR2 DMC.

Chapter 3 *Programmers Model*

Read this for a description of the LPDDR2 DMC memory map and registers.

Appendix A *Signal Descriptions*

Read this for a description of the LPDDR2 DMC signals.

Appendix B *Revisions*

Read this for a description of the technical changes between released issues of this book.

Glossary Read this for definitions of terms used in this book.

Conventions

Conventions that this book can use are described in:

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

Typographical

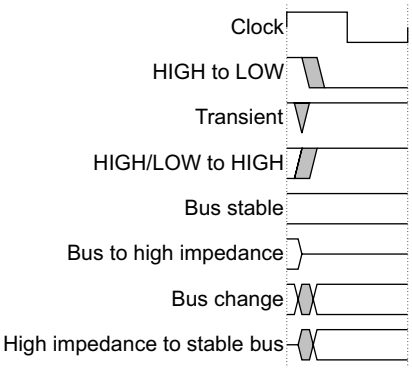
The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes 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.
`mono`space	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 >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

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.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



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
- LOW for active-LOW signals.

Lower-case n At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, http://infocenter.arm.com, for access to ARM documentation.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *AMBA LPDDR2 Dynamic Memory Controller DMC-342 Implementation Guide* (ARM DII 0214)
- *AMBA LPDDR2 Dynamic Memory Controller DMC-342 Integration Manual* (ARM DII 0215)
- *AMBA LPDDR2 Dynamic Memory Controller DMC-342 Supplement to AMBA Designer (FD001) User Guide* (ARM DSU 0012)
- *AMBA Designer (FD001) User Guide* (ARM DUI 0333)
- *AMBA AXI Protocol Specification* (ARM IHI 0022)
- *AMBA 3 APB Protocol Specification (ARM IHI 0024)*
- *AMBA LPDDR2 Dynamic Memory Controller DMC-342 Release Note.*

Other publications

This section lists relevant documents published by third parties:

- *DDR PHY Interface (DFI) Specification*, http://www.ddr-phy.org
- *JEDEC STANDARD Low Power Double Data Rate (LPDDR) SDRAM Specification*, JESD209, http://www.jedec.org
- *JEDEC STANDARD LPDDR2 SDRAM Specification*, JESD209-2, http://www.jedec.org.

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

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

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- the title
- the number, ARM DDI 0436A
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the LPDDR2 DMC and contains the following sections:

1.1 About the AMBA LPDDR2 DMC-342

The LPDDR2 DMC is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM.

The LPDDR2 DMC is a high-performance, area-optimized LPDDR and LPDDR2 SDRAM memory controller compatible with the AMBA AXI protocol.

Figure 1-1 shows an example system.

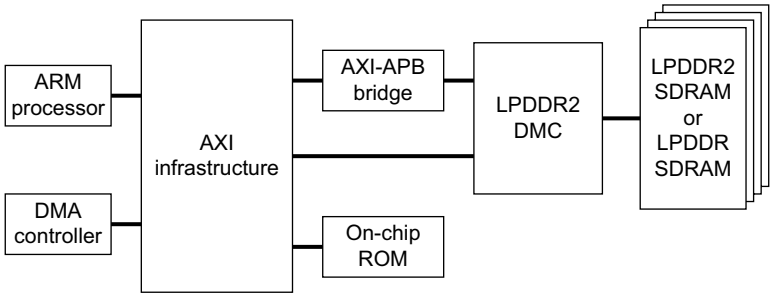


Figure 1-1 Example system

1.2 Compliance

The LPDDR2 DMC is compliant with the following standards and protocols:

- AMBA AXI protocol
- AMBA 3 APB protocol
- JEDEC LPDDR2 standard.

Note

- a memory burst length of 16 is not supported, see Table 2-1 on page 2-11
- supports 333MHz for LPDDR2-S2, and LPDDR2-S4 devices.

1.3 Features

The LPDDR2 DMC features are:

- supports LPDDR SDRAM, LPDDR2-S2, and LPDDR2-S4 devices
- supports the variants (A and B) of LPDDR2 -S2, and LPDDR2-S4 devices
- soft macrocell available in Verilog
- synchronous n:1 clocking between AXI and APB
- synchronous operation between AXI bus infrastructure and external memory bus using clock ratios of 1:1, 1:n, or n:1
- asynchronous operation between AXI bus infrastructure and external memory bus
- active and precharge power-down supported in the memory devices
- interfaces to a PHY device using a *DDR PHY Interface* (DFI) pad interface
- *Quality of Service* (QoS) features for low latency transfers
- optimized utilization of external memory bus
- programmable selection of external memory width, see *Supported memory and AXI data widths* on page 2-11
- multiple outstanding transactions
- hardware resource that can be rendered to optimize area versus performance
- support for a configurable number of ARMv6 architecture outstanding exclusive access transfers
- each LPDDR2 DMC can be configured to have between 1 and 4 memory chip selects.

1.4 Interfaces

The LPDDR2 DMC has the following external interfaces:

- AMBA AXI slave interface, for transfer of memory data to or from an AMBA master
- AMBA 3 APB slave interface, for programming the LPDDR2 DMC
- DFI pad interface, for accesses to the memory devices.

1.5 Configurable options

The LPDDR2 DMC has the following configurable options:

- AXI data width
- memory data width
- number of memory devices
- arbiter depth
- number of exclusive access monitors
- independently controlled **dfi_cke** signals for each chip select or globally controlled signals
- read *First In First Out* (FIFO) depth
- *Read IDentification* (RID) *First In First Out* (FIFO) depth
- write data buffer depth
- command FIFO depth
- *AXI IDentification* (AID) width
- memory banks per chip
- bus width for the **user_status**, **user_config0**, and **user_config1** signals.

1.6 Test features

The LPDDR2 DMC provides:

- scan test signals to enable *Automatic Test Pattern Generation* (ATPG) testing, see *Scan test* on page A-5
- integration test logic, see *Integration test* on page 3-6.

1.7 Product documentation, design flow, and architecture

This section describes the LPDDR2 DMC books, how they relate to the design flow, and the relevant architectural standards and protocols.

For more information about the books described in this section, see *Additional reading* on page xvii.

1.7.1 Documentation

The LPDDR2 DMC documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the LPDDR2 DMC. It is required at all stages of the design flow. Some behavior described in the TRM might not be relevant because of the way that the LPDDR2 DMC is implemented and integrated. If you are programming the LPDDR2 DMC then contact:

- the implementer to determine the build configuration of the implementation
- the integrator to determine the signal configuration of the SoC that you are using.

The TRM complements protocol specifications and relevant external standards. It does not duplicate information from these sources.

User Guide

The *User Guide* (UG) describes:

- the available build configuration options and related issues in selecting them
- how to use AMBA Designer to:
  - configure the LPDDR2 DMC
  - generate the *Register Transfer Level* (RTL).

The UG is a confidential book that is only available to licensees.

Implementation Guide

The *Implementation Guide* (IG) describes the synthesis constraints.

The ARM product deliverables include reference scripts and information about using them to implement your design.

The IG is a confidential book that is only available to licensees.

Integration Manual

The *Integration Manual* (IM) describes how to integrate the LPDDR2 DMC into a SoC. It includes describing the signals that the integrator must tie off to configure the macrocell for the required integration. Some of the integration is affected by the configuration options used when implementing the LPDDR2 DMC.

The IM is a confidential book that is only available to licensees.

1.7.2 Design flow

The LPDDR2 DMC is delivered as synthesizable RTL. Before it can be used in a product, it must go through the following process:

1. Implementation. The implementer configures and synthesizes the RTL to produce a hard macrocell.
2. Integration. The integrator connects the implemented design into an SoC. This includes connecting it to a memory system and peripherals.
3. Programming. The system programmer develops the software required to control the LPDDR2 DMC and tests the required application software.

Each stage of the process:

- can be performed by a different party
- can include options that affect the behavior and features at the next stage:

  Build configuration

  The implementer chooses the options that affect how the RTL source files are pre-processed. They usually include or exclude logic that can affect the area or maximum frequency of the resulting macrocell.

  Configuration inputs

  The integrator configures some features of the LPDDR2 DMC by tying inputs to specific values. These configurations affect the start-up behavior prior to the software taking control. They can also limit the options available to the software. See *Tie-offs* on page A-3.

  Software control

  The programmer updates the LPDDR2 DMC by programming particular values into software-visible registers. This affects the behavior of the LPDDR2 DMC.

1.7.3 ARM architecture and protocol information

The LPDDR2 DMC complies with, or implements, the ARM specifications described in *Advanced Microcontroller Bus Architecture*.

Advanced Microcontroller Bus Architecture

The LPDDR2 DMC complies with the:

- AMBA AXI protocol, see the *AMBA AXI Protocol Specification*
- AMBA 3 APB protocol, see the *AMBA 3 APB Protocol Specification*.

1.8 Product revisions

This section describes the differences in functionality between product revisions of the LPDDR2 DMC:

r0p0 First release.

Chapter 2
Functional Description

This chapter describes the LPDDR2 DMC operation. It contains the following sections:

2.1 About the functions

The LPDDR2 DMC is a configurable memory controller for use with LPDDR SDRAMs and LPDDR2 SDRAMs.

Figure 2-1 shows a block diagram of the LPDDR2 DMC.

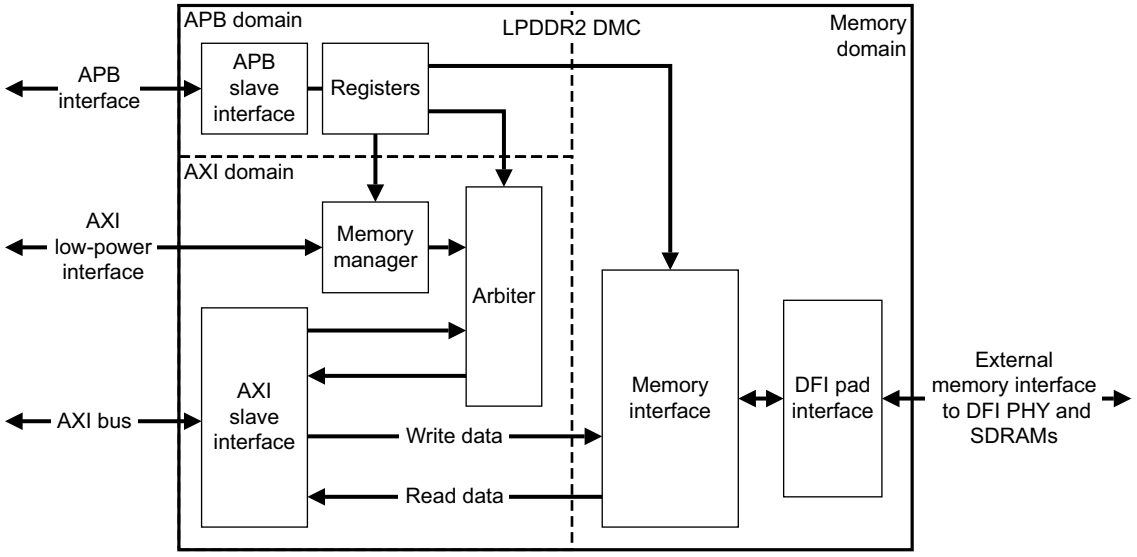


Figure 2-1 Block diagram

2.2 Interfaces

Figure 2-2 shows the interfaces of the LPDDR2 DMC.

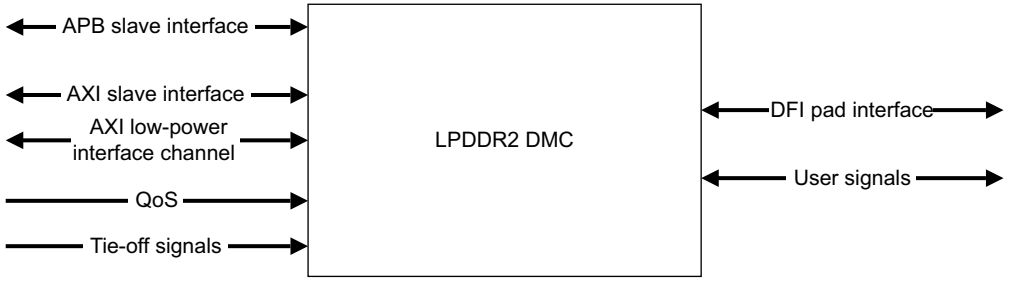


Figure 2-2 LPDDR2 DMC interfaces

The interfaces of the LPDDR2 DMC are:

- *AXI slave interface*
- *AXI low-power interface* on page 2-6
- *APB slave interface* on page 2-6
- *Tie-off signals* on page 2-6
- *User signals* on page 2-7
- *Memory interface* on page 2-7
- *DFI pad interface* on page 2-8
- *QoS signal* on page 2-8.

2.2.1 AXI slave interface

The AXI slave interface comprises the following AXI channels:

Write address Enables the transfer of the address and all other control data required for the LPDDR2 DMC to carry out an AXI write transaction.

Write data Enables the transfer of write data and validating data byte strobes to the LPDDR2 DMC.

Write response Enables the transfer of response information associated with a write transaction.

Read address Enables the transfer of the address and all other control data required for the LPDDR2 DMC to carry out an AXI read transaction.

Read data Enables the transfer of read data and response information associated with a read transaction.

For more information, see the *AMBA AXI Protocol Specification*.

Figure 2-3 on page 2-5 shows the AXI slave interface signals.

Note

In Figure 2-3 on page 2-5 the **arcache**, **awcache**, **arprot**, and **awprot** signals are shown for completeness only. The LPDDR2 DMC ignores any information that these signals provide.

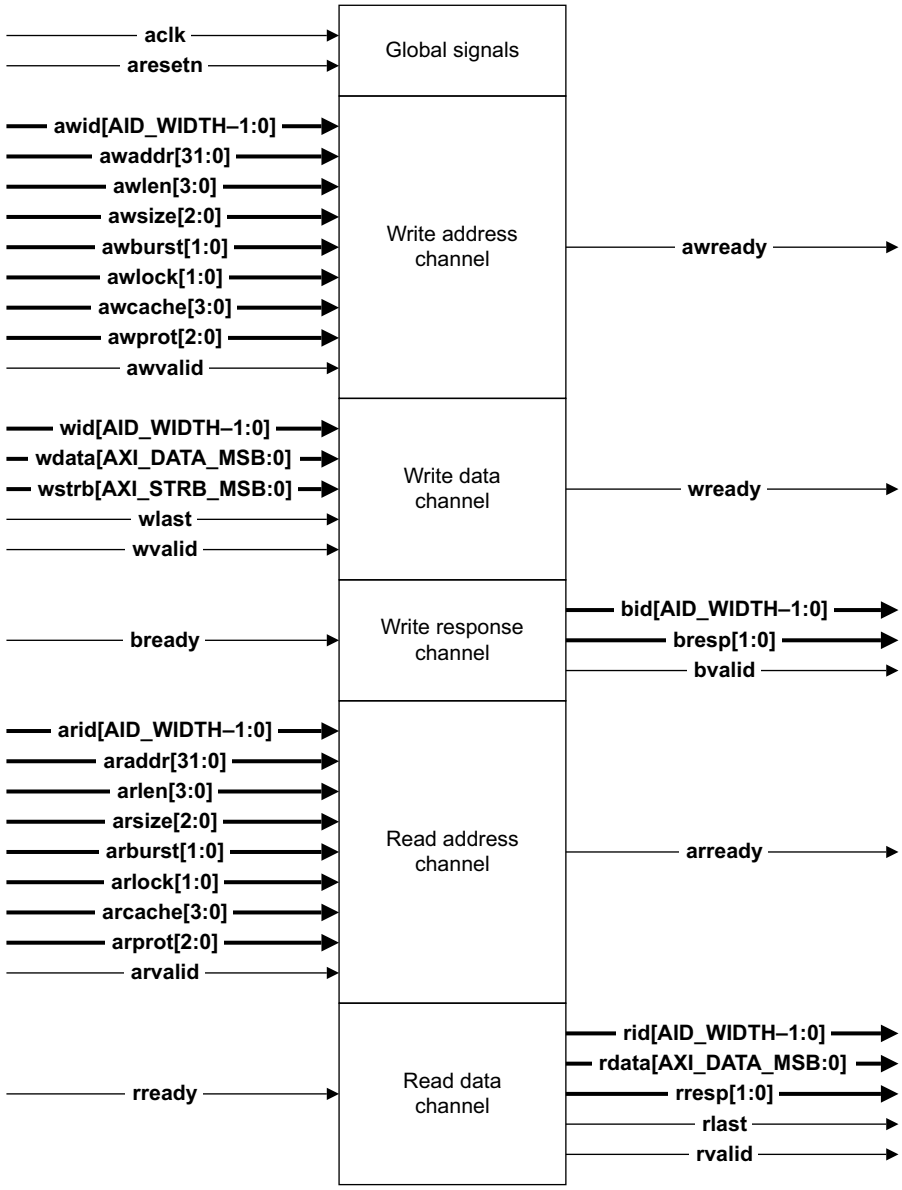


Figure 2-3 AXI slave interface signals

2.2.2 AXI low-power interface

Figure 2-4 shows the AXI low-power interface channel signals.

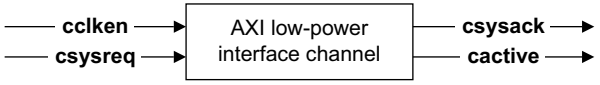


Figure 2-4 AXI low-power interface channel signals

For more information, see *AXI low-power interface* on page 2-14.

2.2.3 APB slave interface

The APB slave interface provides access to the internal registers of the LPDDR2 DMC. Figure 2-5 shows the APB interface signals.

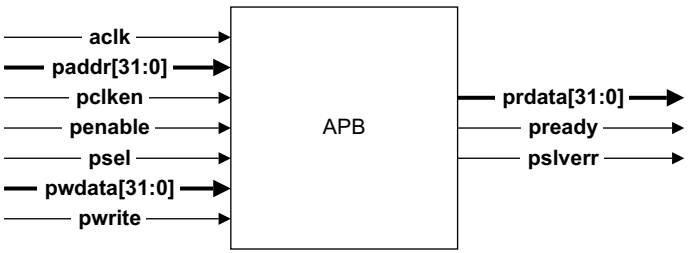


Figure 2-5 APB interface

For more information, see *APB slave interface* on page 2-14.

2.2.4 Tie-off signals

The tie-off signals initialize various operating parameters of the LPDDR2 DMC when it exits the reset state. Figure 2-6 shows the tie-off signals.

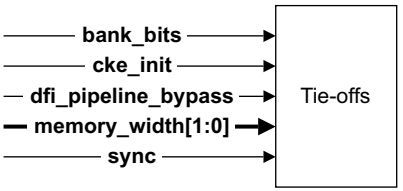


Figure 2-6 Tie-off signals

For more information, see *Tie-off signals* on page 2-15.

2.2.5 User signals

The user signals are general purpose input and output signals that you can control, and monitor by using the registers that the LPDDR2 DMC provides. Figure 2-7 shows the user signals.

Figure 2-7 User signals

For more information, see *Miscellaneous signals* on page 2-16.

2.2.6 Memory interface

The memory interface provides a clean and defined interface between the DFI pad interface and the arbiter, ensuring that the external memory interface command protocols are met in accordance with the programmed timings in the register block. See Chapter 3 *Programmers Model*.

The external inputs and outputs to this block are:

mclk Clock for **mclk** domain.

mresetn Reset for **mclk** domain. This signal is active LOW.

The memory interface tracks and controls the state of a memory device by using an **mclk** FSM. See Figure 2-8.

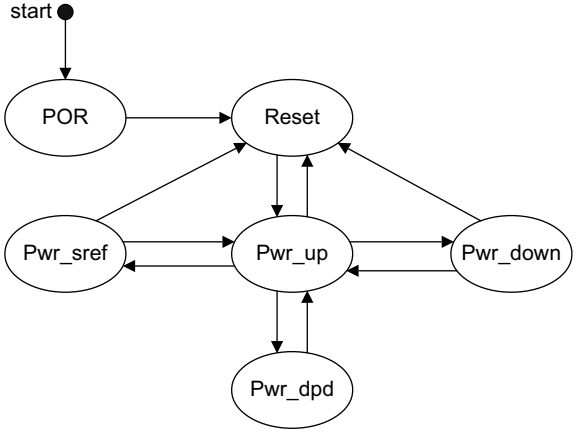


Figure 2-8 mclk domain state diagram

For more information, see *Memory interface* on page 2-23.

2.2.7 DFI pad interface

Figure 2-9 shows the DFI pad interface signals.

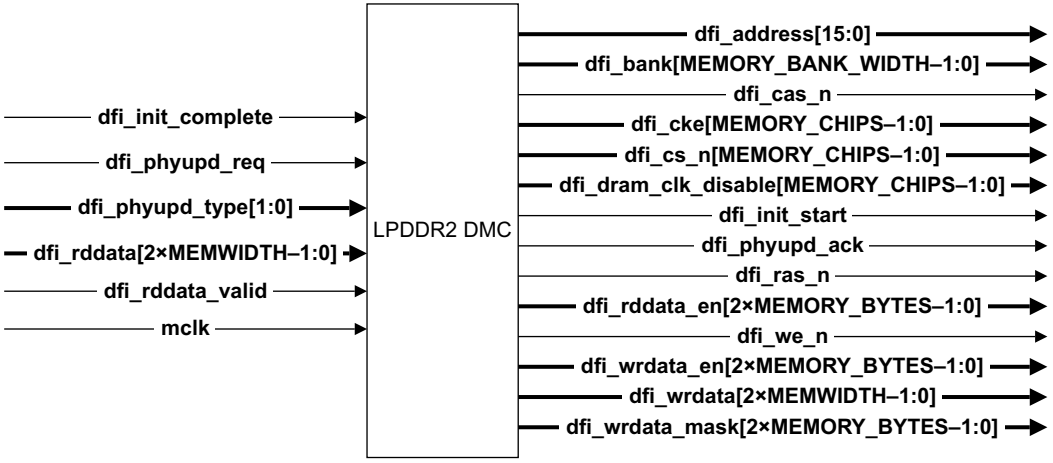


Figure 2-9 DFI pad interface signals

For more information, see *DFI pad interface* on page 2-28.

2.2.8 QoS signal

An AMBA master can use the QoS signal to request that the LPDDR2 DMC assigns minimum latency to the current read transaction. Figure 2-10 shows the QoS signal.

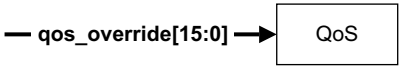


Figure 2-10 QoS signal

For more information, see *Quality of Service* on page 2-18.

2.3 Clocking and resets

This section describes:

- *Clocking*
- *Reset*.

2.3.1 Clocking

The LPDDR2 DMC has the following functional clock inputs:

aclk The **aclk** domain signals can only be stopped if the external memories are put in self-refresh mode.

mclk The **mclk** signal must be clocked at the rate of the external memory clock speed. The **mclk** domain signals can only be stopped if the external memories are put in self-refresh mode.

Note

The LPDDR2 DMC provides the **dfi_dram_clk_disable[MEMORY_CHIPS–1:0]** signal to enable the DFI PHY to disable the clock to a memory device.

2.3.2 Reset

The LPDDR2 DMC has the following reset inputs:

aresetn This is the reset signal for the **aclk** domain.

mresetn This is the reset signal for the **mclk** domain.

You can change both reset signals asynchronously to their respective clock domain. Internally to the LPDDR2 DMC, the deassertion of:

- **aresetn** is synchronized to **aclk**
- **mresetn** is synchronized to **mclk**.

2.4 Operation

This section describes:

- *AXI slave interface*
- *AXI low-power interface* on page 2-14
- *APB slave interface* on page 2-14
- *Tie-off signals* on page 2-15
- *Miscellaneous signals* on page 2-16
- *Arbiter* on page 2-16
- *Memory manager* on page 2-20
- *Memory interface* on page 2-23
- *DFI pad interface* on page 2-28
- *Initialization* on page 2-29
- *Power-down support* on page 2-34
- *Power-down usage model* on page 2-38.

2.4.1 AXI slave interface

The AXI programmer's view is of a flat area of memory. The full range of AXI operations are supported.

The base address of each external memory device is programmable using the appropriate chip_cfg<n> Register. See *Chip Configuration registers* on page 3-45.

In addition to reads and writes, exclusive reads and writes are supported, see *AMBA AXI Protocol Specification*. Successful exclusive accesses have an EXOKAY response. All other accesses, including exclusive fail accesses, receive an OKAY response.

Note

Refreshes can be missed if **rready** is held LOW for longer than one refresh period and the read data FIFO, command FIFO, and arbiter queue become full. An OVL error is triggered if this occurs in simulation. Ensure that the LPDDR2 DMC has a sufficiently high system priority to prevent this.

This section describes:

- *Supported memory and AXI data widths* on page 2-11
- *Write data merging* on page 2-11
- *Early BRESP* on page 2-12
- *AXI slave interface attributes* on page 2-12
- *Formatting from AXI address channels* on page 2-12

- *Exclusive access* on page 2-13.

Supported memory and AXI data widths

The read data FIFO width and write data buffer width are equal to the greater of the AXI or the effective memory data width.

Table 2-1 shows the supported combinations of memory and AXI bus widths.

Table 2-1 Supported combinations of memory and AXI data widths

AXI data width	Memory data width	Effective memory data width[a]	Burst length[b]
32-bit	16-bit	32-bit	4 or 8
32-bit	32-bit	64-bit	4 or 8
64-bit	16-bit	32-bit	4 or 8
64-bit	32-bit	64-bit	4 or 8
128-bit	16-bit	32-bit	8
128-bit	32-bit	64-bit	4 or 8

a. Effective memory data width is equal to the size of transfer on a per-cycle basis on the memory interface.
b. Program this value by using the memory_burst field in the *Memory Configuration Register* on page 3-16.

Note

- In addition to the choice of memory data widths at render time, you can modify the LPDDR2 DMC to use half the configured memory width by either:
  - using the **memory_width[1:0]** tie-off signals, see *Tie-offs* on page A-3
  - programming the memory_width2 field in the *Memory Configuration 2 Register* on page 3-29.
- When modifying the configured memory width you must ensure that:
  - the new memory width is not less than 16 bits
  - if burst length is four then the effective memory width is not less than half the AXI slave interface data width.

Write data merging

The LPDDR2 DMC merges AXI write data to create optimal memory bursts. It can also reorder writes to optimize the utilization of the memory interface.

Early BRESP

To enable early write response timing, the LPDDR2 DMC employs write data buffering and issues the **BRESP** transfer before the data has been committed to the memory device. The response is sent after the last data beat is accepted by the AXI slave interface and stored in the write data buffer. The controller maintains data coherency for read after write hazards, see *Hazard detection* on page 2-17.

You can enable or disable the early write response feature by using the *Feature Control Register* on page 3-49.

Note

For exclusive write accesses, the controller only issues a **BRESP** transfer after the write transaction is committed to a memory device.

AXI slave interface attributes

Table 2-2 shows the AXI slave attributes and their values.

Table 2-2 AXI slave interface attributes

Attribute[a]	Value
Combined acceptance capability	Arbiter queue depth
Write interleave depth	1
Read data reorder depth	Combined acceptance capability

a. For a description of these AXI attributes, see *Glossary*.

Formatting from AXI address channels

Formatting is as follows:

Chip select decoding

The LPDDR2 DMC compares an incoming address on **araddr[31:24]**, or **awaddr[31:24]**, with each address_match field programmed in the chip_cfg<n> registers. When a match occurs the LPDDR2 DMC asserts chip select <n>. The address_mask field enables you to exclude some bits from the address comparison. See *Chip Configuration registers* on page 3-45.

If no address match occurs then the LPDDR2 DMC still performs the transfer but the result is undefined.

Row select decoding

The row address is determined from the AXI address using the row_bits field in the memory_cfg Register, and also the brc_n_rbc bit for the selected chip defined in the chip_cfg<n> Register. See *Memory Configuration Register* on page 3-16 and *Chip Configuration registers* on page 3-45.

Column select decoding

The column address is determined from the AXI address using the column_bits field of the memory_cfg Register. See *Memory Configuration Register* on page 3-16.

Bank select decoding

The chip bank is determined from the AXI address using the bank_bits field of the memory_cfg2 Register, and also the brc_n_rbc bit for the selected chip defined in the chip_cfg<n> Register. See *Memory Configuration 2 Register* on page 3-29 and *Chip Configuration registers* on page 3-45.

Number of beats

The number of memory beats is determined, depending on the effective external memory width and the burst size of the AXI access. AXI wrapping bursts are split into two incrementing bursts. AXI fixed bursts are split into AXI burst length memory bursts.

Note

On the memory interface, the LPDDR2 DMC supports a burst length of four or eight.

Exclusive access

In addition to reads and writes, the LPDDR2 DMC supports exclusive reads and writes, see the *AMBA AXI Protocol Specification*.

Successful exclusive accesses have an EXOKAY response. All other accesses, including exclusive fail accesses, receive an OKAY response.

Exclusive access monitors implement the exclusive access functionality. Each monitor can track a single exclusive access. The number of monitors is a configurable option.

If an exclusive write fails, the LPDDR2 DMC forces **dfi_wrdata_mask[]** LOW, so that the data is not written.

When monitoring an exclusive access, the address of any write from another master is compared with the monitored address to check that the location is not being updated.

If the write crosses an address boundary, then all of the least-significant address bits up to that boundary are not compared.

Example 2-1 provides information about three transactions.

Example 2-1 Exclusive accesses

Exclusive Read	Address = `0x000`, size = WORD, length = 1, ID = 0.
Write	Address = `0x004`, size = WORD, length = 2, ID = 1.
Exclusive Write	Address = `0x000`, size = WORD, length = 1, ID = 0.

The LPDDR2 DMC marks the exclusive write as having failed because the write crosses the `0x010` address boundary. The write, for comparison purposes, has therefore accessed the region `0x000-0x01F` because bits 4:0 have not been compared.

The burst type is always taken to be INCR when calculating the address region accessed by the write, Therefore, a wrapped transaction in Example 2-1 that wraps down to `0x0` rather than cross the boundary, is treated in the same way. This is the same for a fixed burst that does not cross the boundary or wrap down to `0x0`.

2.4.2 AXI low-power interface

The low-power interface can move the LPDDR2 DMC into Low_power state without the requirement for any register accesses, see *aclk domain state diagram* on page 2-21 and *Low-power mode* on page 2-35.

For more information about the AXI low-power interface, see the *AMBA AXI Protocol Specification*.

If you do not require the low-power interface, tie it off as *AMBA LPDDR2 Dynamic Memory Controller DMC-342 Integration Manual* describes.

2.4.3 APB slave interface

The APB slave interface is a fully compliant APB slave. For information about an APB slave interface, see the *AMBA 3 APB Protocol Specification*.

The APB interface enables you to access the operating state of the LPDDR2 DMC and to program it with the correct timings and settings for the connected memory device. For more information, see Chapter 3 *Programmers Model*. The APB interface also initializes the connected memory devices, see *Initialization* on page 2-29.

Note

The APB only supports single-word 32-bit accesses. The LPDDR2 DMC ignores **paddr[1:0]**, resulting in byte and halfword accesses being treated as word accesses.

The APB interface is clocked by the same clock as the AXI domain clock, **aclk**. The LPDDR2 DMC provides a clock enable, **pclken**, enabling the APB interface to be slowed down and execute at an integer divisor of **aclk**.

To enable a clean registered interface to the external infrastructure, the APB interface always adds a wait state for all reads and writes by driving **pready** LOW. In the following instances, a delay of more than one wait state can be generated when:

- a direct command is received and there are outstanding commands that prevent a new command being stored in the command FIFO
- a memory command is received and a previous memory command has not completed.

The only registers that can be accessed when the LPDDR2 DMC is not in the Config or Low_power state are:

- memc_status Register. Use this to read the configuration and current state of the controller, see *Memory Controller Status Register* on page 3-10.
- memc_cmd Register. Use this to change the operating state, see *Memory Controller Command Register* on page 3-12.

To guarantee no missed `AUTO REFRESH` commands, it is recommended that any change of **mclk** period, and therefore update of the refresh period, is carried out when the LPDDR2 DMC is in the Low_power state. This is because the refresh rate depends on the **mclk** period. Only write direct commands to the external memories when the LPDDR2 DMC is in the Config state and not in the Low_power state.

2.4.4 Tie-off signals

The LPDDR2 DMC enables you to change some of its configuration settings by using the tie-off signals.

At reset, the value of a tie-off signal controls the operating behavior of the controller and it sets the value of the corresponding field in the memory_cfg2 Register.

After reset you can program the memory_cfg2 Register to make changes to these configuration settings. See *Memory Configuration 2 Register* on page 3-29.

Note

The memory_cfg2 Register does not provide a dfi_pipeline_bypass bit and therefore you cannot program the controller to override the **dfi_pipeline_bypass** tie-off.

2.4.5 Miscellaneous signals

You can use the following general-purpose signals to control or monitor logic that is external to the LPDDR2 DMC:

user_status[USER_STATUS_WIDTH–1:0]

You can read the status of these general-purpose inputs by using the *User Status Register* on page 3-46. You must tie any unused signals to either HIGH or LOW. These signals are connected directly to the APB interface block. Therefore, if they are driven from external logic that is not clocked by the **aclk** signal then external synchronization registers are required.

user_config0[USER_CONFIG0_WIDTH–1:0]

The user_config0 Register controls these general-purpose outputs, see *User Config 0 Register* on page 3-47. If you do not require these signals, leave them unconnected.

user_config1[USER_CONFIG1_WIDTH–1:0]

The user_config1 Register controls these general-purpose outputs, see *User Config 1 Register* on page 3-48. If you do not require these signals, leave them unconnected.

You can use the following miscellaneous signal to change the operational behavior of the LPDDR2 DMC:

rst_bypass Use this signal for *Automatic Test Pattern Generation* (ATPG) testing only. You must tie it LOW for normal operation.

2.4.6 Arbiter

The arbiter receives memory access commands from the AXI slave interface and the memory manager. It passes the highest priority command to the memory interface after arbitration. Read data is passed from the memory interface to the AXI slave interface.

This section describes:

- *Formatting from memory manager* on page 2-17
- *Arbiter access multiplexor* on page 2-17

- *Hazard detection*
- *Scheduler* on page 2-18
- *Quality of Service* on page 2-18
- *Arbitration algorithm* on page 2-20
- *Command formatting* on page 2-20.

Formatting from memory manager

The direct command uses the chip select bits [21:20] in the direct_cmd Register to select the required memory chip. See *Direct Command Register* on page 3-14.

The command to be carried out is either:

- a self-refresh request from the AXI-C interface
- a direct command from the APB interface.

It is encoded by the memory manager to match the format that the arbiter requires.

Arbiter access multiplexor

The selection of a command from the AXI slave interface or the memory manager is fixed, with the memory manager having a higher priority.

The selection between an AXI read access and AXI write access is made using a round-robin arbitration, unless a read access has a low latency QoS value, or if any of the accesses are to a row already open. See *Arbitration algorithm* on page 2-20 for more information.

See also *Formatting from AXI address channels* on page 2-12. In this case, it is arbitrated immediately.

Hazard detection

The following types of hazard exist:

Read After Read **(RAR)**

There is a read already in the arbiter queue with the same ID as the incoming entry, that is also a read.

Write After Write **(WAW)**

There is a write already in the arbiter queue with the same ID as the incoming entry, that is also a write.

***Read After Write* (RAW)**

There is a write in the arbiter queue, that has received an early write response, accessing the same location as the incoming read entry.

The arbiter entry is flagged as having a dependency if a hazard is detected. There might be dependencies against a number of other arbiter entries. As the arbiter entries are invalidated, so the dependencies are reduced until finally, there are no outstanding dependencies, and the entry is free to start.

Note

There are no *Write-After-Read* (WAR) hazard checks in the LPDDR2 DMC. If an AXI master requires ordering between reads and writes to certain memory locations, it must wait for read data before issuing a write to a location it has read from. Similarly, the only RAW hazard checking is that performed when the write response has been issued. If an AXI master required ordering between writes and reads to certain memory locations, it must wait for the write response before issuing the read to the same location.

Scheduler

The scheduler monitors the activity of the **mclk** FSMs in the memory interface. This enables the arbiter to select an entry from the queue that does not stall the memory pipeline.

Quality of Service

Quality of Service (QoS) is defined for the LPDDR2 DMC as a method of increasing the arbitration priority of a read access that requires low-latency read data. The QoS for an AXI read access is determined when it is received by the arbiter.

There is no QoS for write accesses. However, any write access that is a dependency for a QoS read access receives a promoted priority to complete as soon as possible. Dependencies are formed based on the hazard detection logic that *Hazard detection* on page 2-17 describes.

The following sections describe:

- *QoS selection* on page 2-19
- *QoS timeout* on page 2-19.

QoS selection

The LPDDR2 DMC selects the QoS for an AXI read transfer by masking the **arid** with a 4-bit QoS mask. You can program the QoS mask to be either **arid[3:0]**, **arid[4:1]**, **arid[5:2]**, **arid[6:3]**, **arid[7:4]**, **arid[8:5]**, **arid[9:6]**, or **arid[10:7]** by using the qos_master_bits in the memory_cfg Register. After the LPDDR2 DMC applies the QoS mask to the four specified **arid** bits, the resulting value <n> provides the pointer to which id_<n>_cfg Register contains the QoS settings for the read transfer. See *Memory Configuration Register* on page 3-16 and *QoS Configuration registers* on page 3-44.

Example 2-2 shows QoS selection and the impact of the **qos_override** signal.

Example 2-2 QoS selection and qos_override

If you program the qos_master_bits = b010 then this selects **arid[5:2]** to be the 4-bit QoS mask. If the LPDDR2 DMC receives an AXI transfer with an **arid** of `0x5A` then it applies the 4-bit QoS mask, **arid[5:2]**, giving a value of `0x6`. Therefore, the controller uses the id_6_cfg Register to control the QoS for the transfer.

The controller creates a new arbiter entry for the transfer and assigns it the qos_min and qos_max values from the id_6_cfg Register. If the qos_enable bit=1 then the controller applies the QoS settings to the transfer.

The **qos_override[15:0]** signal enables the controller to assign an arbiter entry with minimum QoS latency, irrespective of the state of the qos_enable bit. For this example, if **qos_override[6]** is HIGH when **arvalid** and **arready** are HIGH then the arbiter entry is assigned minimum QoS latency, even if the qos_enable bit=0.

QoS timeout

If the qos_enable bit is set then the arbiter decrements the QoS maximum latency value every **aclk** cycle until it reaches zero. If the entry is still in the queue when the QoS maximum latency value reaches zero, then the entry becomes high priority. This is called a *timeout*. Also, any entry in the queue with a minimum latency QoS also produces a timeout. Minimum latency timeouts have priority over maximum latency timeouts.

When an entry times out in this way it forces a timeout onto any entries that it has dependencies against. In normal operation, these entries have already timed out because they have received the same initial QoS value but been decrementing for longer. The highest priority arbiter entry is serviced next.

One special case exists. This is when or if the assertion of the relevant **qos_override** signal forces a minimum latency timeout. In this instance, any accesses that the new entry has dependencies against might not have timed out and are forced to time out so that the high-priority entry can start as soon as possible.

Arbitration algorithm

The ordering of commands to be carried out from the arbiter queue is arbitrated with a priority scheme of the following order:

- refresh timeouts
- minimum latency timeouts
- maximum latency timeouts
- open-row hits in the same direction
- open-row hits in the opposite direction
- preparation operations
- refreshes
- QoS transactions.

Note

Preparation operations can be interleaved between memory operations to other banks to improve memory interface utilization.

Command formatting

For every memory burst access necessary to complete an arbiter queue entry, a memory interface command is required.

Command formatting calculates the number of memory interface commands and memory cycles for each command to complete the next arbiter queue entry that is to be sent to the memory interface. It contains an address incrementer and a beat decrementer so that the arbiter entry can be interrupted and restarted.

2.4.7 Memory manager

The memory manager tracks and controls the current state of the LPDDR2 DMC using the **aclk** *Finite State Machine* (FSM). You can change the state of the controller by programming the memc_cmd Register, see *Memory Controller Command Register* on page 3-12.

You can also use the AXI low-power interface to move the controller between the Ready and Low_power states, see *Low-power mode* on page 2-35.

Figure 2-11 shows the **aclk** FSM.

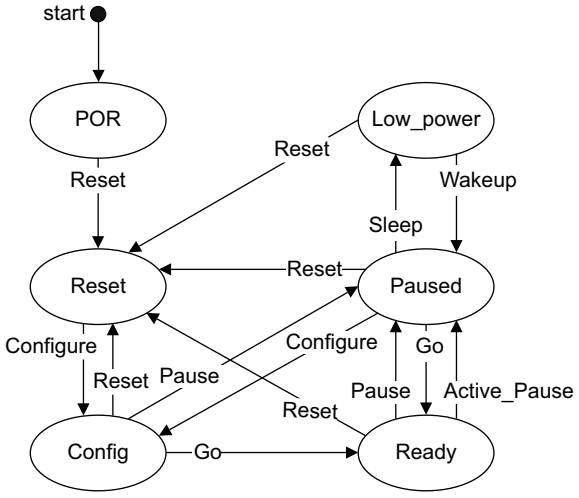


Figure 2-11 aclk domain state diagram

In Figure 2-11, non-state moving transitions are omitted for clarity. See Table 2-8 on page 2-38 for valid system states.

Note

- If the LPDDR2 DMC receives an APB command that is illegal to carry out from the current state then the controller ignores it and the **aclk** FSM stays in the same state.
- For the two cycles following *Power-On Reset* (POR), do not consider the LPDDR2 DMC to be in the Config state. For this reason, register access restrictions apply.
- You can only use the AXI low-power interface to move in and out of the Low_power state from the Ready state.
- If the LPDDR2 DMC enters the Low_power state using the:
  - APB interface then it must also exit the Low_power state using the APB interface
  - AXI low-power interface then it must also exit the Low_power state using the AXI low-power interface.
- When the LPDDR2 DMC is in the Ready state it periodically issues `AUTO REFRESH` commands to the memory devices. When the controller is in the Config or Paused state it does not issue `AUTO REFRESH` commands unless software writes the

appropriate command to the direct_cmd Register. However, this register is only accessible in the Config state and therefore software must limit the duration of the Paused state.

- If you move the LPDDR2 DMC from the Low_power state to the Paused state, then you must not move the controller immediately back to the Low_power state, if it has not sent an AUTO REFRESH command. This is because, when a memory device exits self-refresh mode and then enters self-refresh mode, the *JEDEC STANDARD LPDDR2 SDRAM Specification* requires a controller to issue at least one AUTO REFRESH command.

If a previous command has not completed, then the APB slave interface stalls the **psel** and **penable** signals using the **pready** signal.

The memory manager issues commands from one of the following sources:

Direct commands

These are received over the APB interface as a result of a write to the direct_cmd Register, see *Direct Command Register* on page 3-14. Use these commands to initialize the memory devices and generate periodic refresh commands.

The legal commands that the memory manager uses are:

- NOP
- PRECHARGEALL
- AUTO REFRESH
- MODEREG
- extended MODEREG.
- Deep Power-Down (DPD).

Commands from the aclk FSM

You can traverse the **aclk** FSM by writing to the memc_cmd Register. See *Memory Controller Command Register* on page 3-12. You can only traverse the **aclk** FSM states when the LPDDR2 DMC is idle. For example, the Ready state can only be entered from the Config state when all direct commands have been completed. The exception to this is the Active_Pause command. You can issue this command when the LPDDR2 DMC is active. When you issue the command, any memory accesses that have not been arbitrated remain in the arbiter until the **aclk** FSM receives the Go command.

Refresh commands

The refresh logic can issue commands to the arbiter to refresh the memory devices. The refresh rate period is programmable using the *Refresh Period Register* on page 3-18. The value of this register is the time period in **mclk** cycles that must occur before the memory manager requests the arbiter to generate an AUTO REFRESH command. This request is arbitrated and might not necessarily be initiated immediately. See *Arbiter* on page 2-16.

2.4.8 Memory interface

The memory interface is separated from the arbiter using the following configurable blocks:

- command FIFO
- read data FIFO
- write data buffer.

The memory interface reads commands from the arbiter using the command FIFO but only when that command can be executed. The memory interface ensures a command is only executed when all the inter-command delays, defined in this section, for that bank or chip are met.

The memory interface enables multiple banks to be active at any one time. However, only one bank can be carrying out a data transfer at any one time. If the command at the head of the command FIFO cannot be executed, then the command pipeline stalls until it can be executed.

When the auto_power_down bit is set, see *Memory Configuration Register* on page 3-16, then the **dfi_cke** output signal is negated to take the external memories into active power-down or precharge power-down depending on whether there is a row open. See Figure 2-20 on page 2-27. When exiting power down mode, the delay before the next command is issued is programmed using the *Exit Power-down Timing Register* on page 3-27.

An **mclk** FSM controls the operation of the power-down mode. This FSM has a state that is entered when its memory device is put into self-refresh mode. This is used so that if power is removed from all of the LPDDR2 DMC apart from the memory interface and DFI pad interface, the state of the memory is known. When the rest of the LPDDR2 DMC is powered-up, the **aclk** FSM enters the Low_power state rather than the Config state.

Memory interface to DFI pad interface timing

All command control outputs are clocked on the same edge of the memory clock, **mclk**.

The relative times between control signals from the memory interface are maintained when output from the DFI pad interface to the actual memory devices. Therefore, the timing register values required for a particular memory device can be determined from the memory device's data sheet.

Figure 2-12 to Figure 2-23 on page 2-28 show how the data sheet timings map onto the LPDDR2 DMC timing registers. The timing registers are shown in Figure 3-2 on page 3-4.

Note

In Figure 2-12 to Figure 2-23 on page 2-28:

- The following signals are internal to the LPDDR2 DMC:
  - **command_en**
  - **memif_busy**
  - **pwr_down**.
- The timings shown are not necessarily the default timing values but are values that are small enough to show the entire delay in one figure.

Figure 2-12 shows the `ACTIVE` command to `Read` or `Write` command timing, that you program using the *ACTIVE to Read or Write Timing Register* on page 3-23.

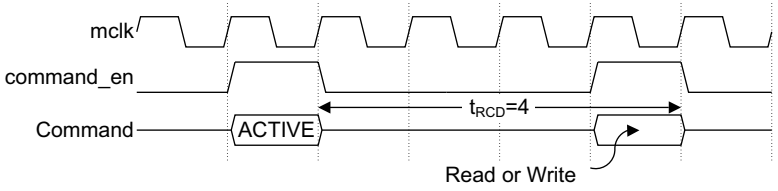


Figure 2-12 `ACTIVE` command to `Read` or `Write` command timing, t_{RCD}

Figure 2-13 on page 2-25 shows the four activate window command timing, that you program using the *Four Activate Window Timing Register* on page 3-33.

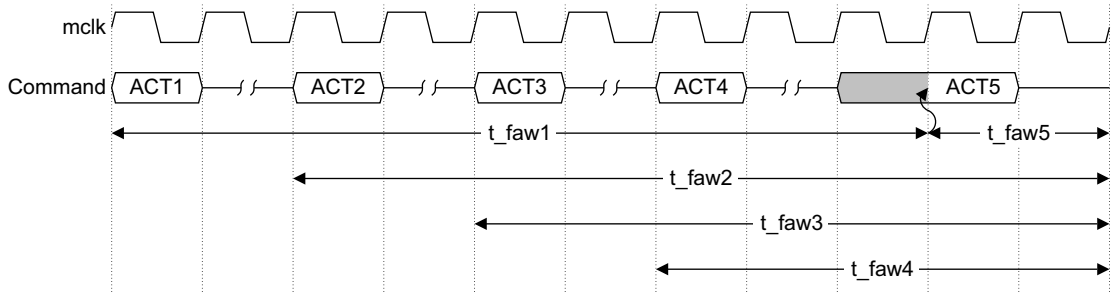


Figure 2-13 Four activate window command timing, t_{FAW}

Figure 2-14 shows ACTIVE to ACTIVE on the same bank, and ACTIVE to AUTO REFRESH command timing, that you program using the *ACTIVE to ACTIVE Timing Register* on page 3-22.

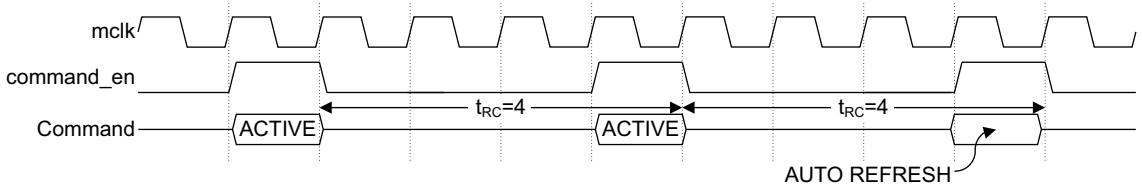


Figure 2-14 Same bank ACTIVE to ACTIVE, and ACTIVE to AUTO REFRESH command timing, t_{RC}

Figure 2-15 shows the ACTIVE to ACTIVE command timing to different memory banks, that you program using the *ACTIVE to ACTIVE Different Bank Timing Register* on page 3-25.

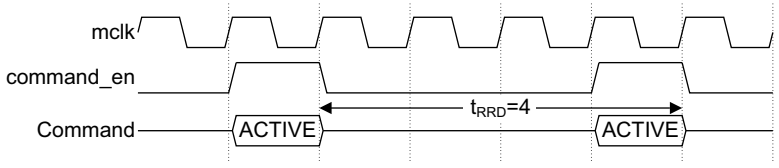


Figure 2-15 Different bank ACTIVE to ACTIVE command timing, t_{RRD}

Figure 2-16 on page 2-26 shows the PRECHARGE to command, and AUTO REFRESH timing, that you program using the *PRECHARGE to Command Timing Register* on page 3-24 and *AUTO REFRESH to Command Timing Register* on page 3-23.

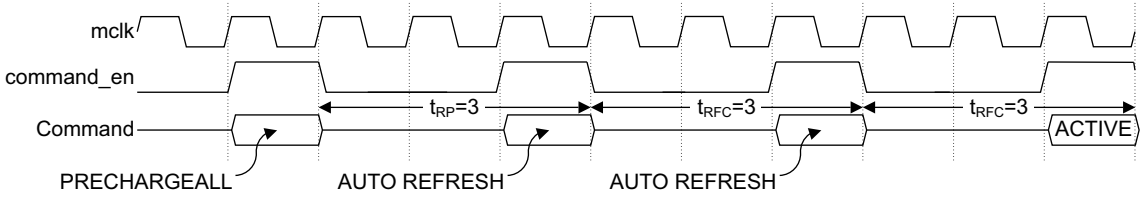


Figure 2-16 PRECHARGE **to command and** AUTO REFRESH **to command timing, t_{RP} and t_{RFC}**

Figure 2-17 shows ACTIVE to PRECHARGE, and PRECHARGE to PRECHARGE timing, that you program using the *ACTIVE to PRECHARGE Timing Register* on page 3-21 and *PRECHARGE to Command Timing Register* on page 3-24.

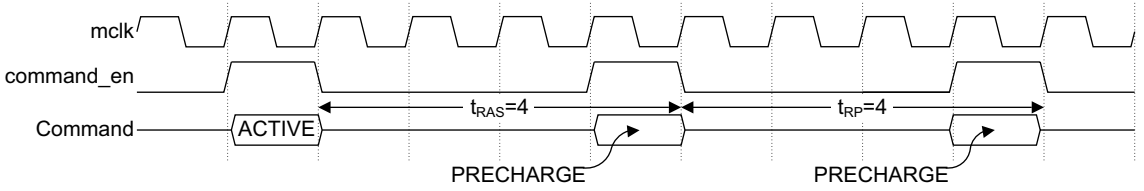


Figure 2-17 ACTIVE **to** PRECHARGE**, and** PRECHARGE **to** PRECHARGE **timing, t_{RAS} and t_{RP}**

Figure 2-18 shows MODEREG to command timing, that you program using the *MODEREG to Command Timing Register* on page 3-20.

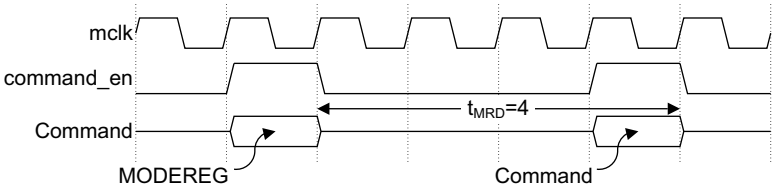


Figure 2-18 MODEREG **to command timing, t_{MRD}**

Figure 2-19 on page 2-27 shows self-refresh entry and exit timing, that you program using the *Self-refresh to Command Timing Register* on page 3-28 and *Exit Self-refresh Timing Register* on page 3-28.

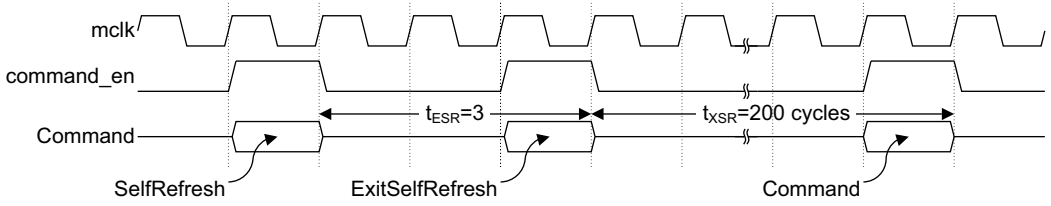


Figure 2-19 Self-refresh entry and exit timing, t_{ESR} and t_{XSR}

Figure 2-20 shows power-down entry and exit timing, that you program using the *Memory Configuration 3 Register* on page 3-32 and *Exit Power-down Timing Register* on page 3-27. It also shows the clock enable timing that you program using the *Clock Enable Register* on page 3-38.

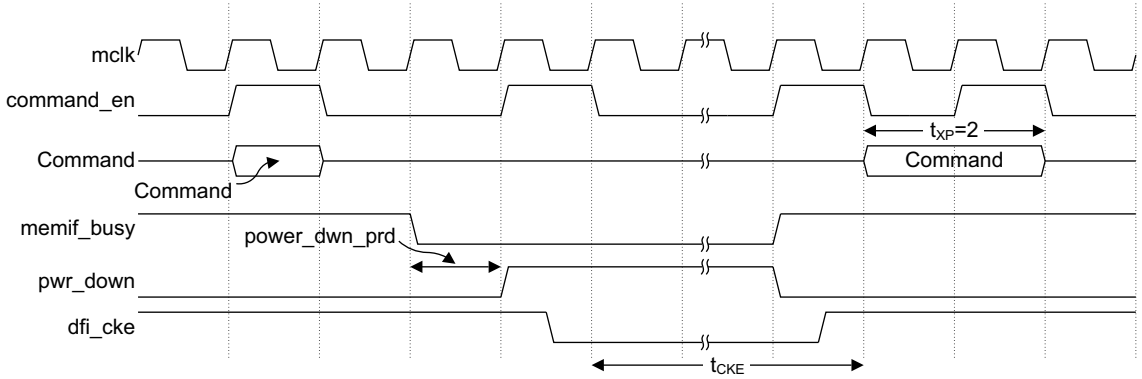


Figure 2-20 Power-down entry and exit timing, t_{XP}

The power_dwn_prd count is timed from the memory interface becoming idle, that is, after a command delay has timed out or the read data FIFO is emptied. **dfi_cke** is asserted when the command FIFO is not empty.

Figure 2-21 on page 2-28 shows the relationship between the memory interface outputting a `Write` command and the write data, that you program using the *Write Latency Register* on page 3-20. It also shows the turnaround time, t_{WTR}, for the memory interface to output a `Write` command followed immediately by a `Read` command. Program t_{WTR} by using the *Write to Read Timing Register* on page 3-26.

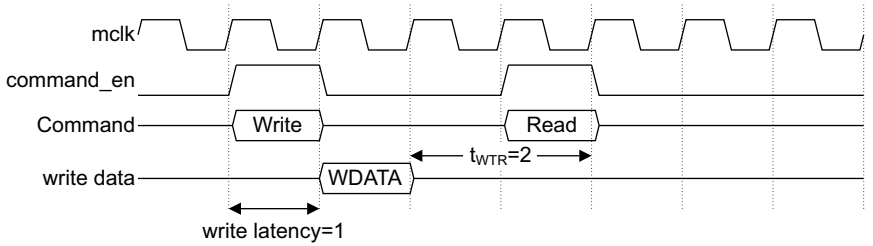


Figure 2-21 Data output timing, t_{WTR}

Figure 2-22 shows the t_{WR} minimum time between a `Write` and a `PRECHARGE` command, that you program using the *Write to PRECHARGE Timing Register* on page 3-26.

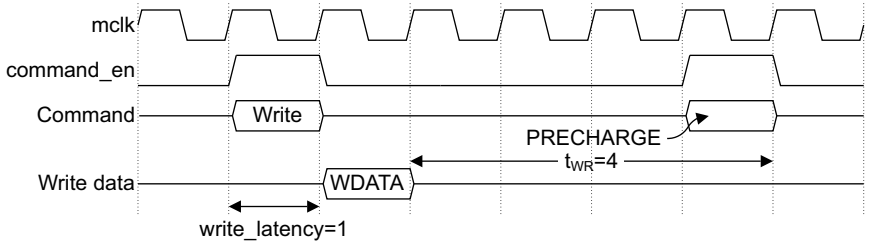


Figure 2-22 `Write` to `PRECHARGE` timing, t_{WR}

Figure 2-23 shows the timing relationship between a `Read` command being output from the memory interface and the read data being returned to the memory interface from the DFI pad interface. Program t_{rddata_en} by using the *Read Data Enable Timing Register* on page 3-35.

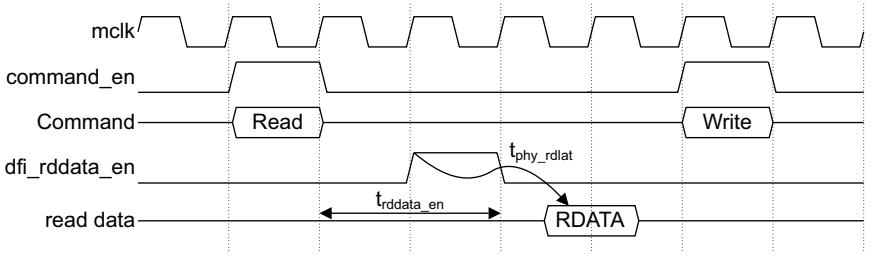


Figure 2-23 Data input timing

2.4.9 DFI pad interface

The LPDDR2 DMC implements a DFI pad interface that complies with Version 2.1 of the *DDR PHY Interface (DFI) Specification*.

2.4.10 Initialization

Before you can use the LPDDR2 DMC operationally to access external memory, you must move it to the Config state and then:

- program the LPDDR2 DMC configuration and timing registers
- initialize the external memory devices by programming the direct_cmd Register.

You might not have to program all of the LPDDR2 DMC registers because some might reset to the correct value.

Note

A deadlock situation might occur if an AXI master accesses the AXI slave interface before a master has programmed the LPDDR2 DMC using the APB interface. Before the LPDDR2 DMC has been initialized, if a master can access the AXI slave interface but cannot access the APB interface, then that master must be held off until another master programs the LPDDR2 DMC.

The following sections describe:

- *Controller initialization*
- *LPDDR device initialization* on page 2-32
- *LPDDR2-S2 device initialization* on page 2-33.

Controller initialization

Table 2-3 shows an example initialization programming sequence for the controller.

Table 2-3 Controller initialization example

Register	Write data	Description
cas_latency	0x00000006	Set CAS latency to 3
t_mrd	0x00000002	Set t_{MRD} to 2
t_ras	0x00000007	Set t_{RAS} to 7
t_rc	0x0000000B	Set t_{RC} to 11
t_rcd	0x00000305	Set t_{RCD} to 5 and schedule_rcd to 3
t_rfc	0x0000282A	Set t_{RFC} to 42 and schedule_rfc to 40
t_rp	0x00000103	Set t_{RP} to 3 and schedule_rp to 1
t_rrd	0x00000002	Set t_{RRD} to 2

Table 2-3 Controller initialization example (continued)

Register	Write data	Description
t_wr	0x00000003	Set t_{WR} to 3
t_wtr	0x00000002	Set t_{WTR} to 2
t_xp	0x00000001	Set t_{XP} to 1
t_xsr	0x000000C8	Set t_{XSR} to 200
t_esr	0x00000003	Set t_{ESR} to 3
t_rddata_en	0x00000003	Set t_{rddata_en} to 3
refresh_prd	0x000003D0	Set an auto-refresh period of 2.93µs, that is, every 976 (or 0x3D0) **mclk** periods when **mclk** frequency is 333MHz
chip_cfg0	0x000000FF	Sets address for chip 0 to be 0x00XXXXXX, *Row Bank Column* (RBC) configuration
chip_cfg1	0x000022FF	Sets address for chip 1 to be 0x22XXXXXX, RBC configuration
chip_cfg2	0x000055FF	Sets address for chip 2 to be 0x55XXXXXX, RBC configuration
chip_cfg3	0x00007FFF	Sets address for chip 3 to be 0x7FXXXXXX, RBC configuration
memory_cfg	-	Write to the memory configuration registers, see either: • Table 2-4 on page 2-31 • Table 2-5 on page 2-32.
memory_cfg2	-	
direct_cmd	-	Sequence of writes to initialize the memory devices, see either: • *LPDDR device initialization* on page 2-32 • *LPDDR2-S2 device initialization* on page 2-33
memc_cmd	0x00000000	Write the Go command to move the controller to the Ready state
memc_status	-	Poll the register until the memc_status field returns b01, signifying that the controller is ready to accept AXI accesses to the memory devices

Table 2-4 shows example settings of the memory_cfg and memory_cfg2 registers for use with LPDDR devices.

Table 2-4 memory_cfg and memory_cfg2 register example, for LPDDR devices

Register	Write data	Description
memory_cfg	`0x00610011`	Sets the following memory configuration: • 9 column bits, 13 row bits • power-down period of 0 • disable auto power-down • disable dynamic clock stopping • memory burst size of 4 • use **arid[3:0]** bits for QoS • four memory devices.
memory_cfg2	`0x00000229`	Sets the following memory configuration: • **aclk** and **mclk** are synchronous • **dfi_cke** is HIGH at reset • two bits for the bank address • 16-bit data bus • memory type is LPDDR.

Table 2-5 shows example settings of the memory_cfg and memory_cfg2 registers for use with LPDDR2-S2 devices.

Table 2-5 memory_cfg and memory_cfg2 Register example, for LPDDR2-S2 devices

Register	Write data	Description
memory_cfg	0x00610023	Sets the following memory configuration: • 11 column bits, 15 row bits • power-down period of 0 • disable auto power-down • disable dynamic clock stopping • memory burst size of 4 • use **arid[3:0]** bits for QoS • four memory devices.
memory_cfg2	0x00000721	Sets the following memory configuration: • **aclk** and **mclk** are synchronous • **dfi_cke** is LOW • two bits for the bank address • 16-bit data bus • memory type is LPDDR2-S2.

LPDDR device initialization

Table 2-6 shows an example programming sequence for LPDDR device initialization.

Table 2-6 LPDDR device initialization example

Register	Write data	Description
direct_cmd	0x000C0000	NOP command to chip 0
direct_cmd	0x00000000	PRECHARGEALL command to chip 0
direct_cmd	0x00040000	AUTO REFRESH command to chip 0
direct_cmd	0x00040000	AUTO REFRESH command to chip 0
direct_cmd	0x00080032	MODEREG command, with low address bits = 0x32, to chip 0
direct_cmd	0x00090000	Extended MODEREG command, with low address bits = 0x0, to chip 0

If a configured LPDDR2 DMC supports more than one memory chip then repeat the sequence in Table 2-6 on page 2-32 but update the chip_addr field to select each additional memory chip.

LPDDR2-S2 device initialization

Table 2-7 shows an example programming sequence for LPDDR2-S2 device initialization.

Table 2-7 LPDDR2-S2 device initialization example

Register	Write data	Description
direct_cmd	0x000C0000	NOP command to chip 0.
Wait 200µs to enable memory devices to stabilize.		
direct_cmd	0x0008003F	MODEREG write to mode register 63, to reset chip 0.
direct_cmd	0x00880000	MODEREG read of mode register 0, for chip 0.
mrr_data	-	Returns 0x00000000? When 0x0 is returned, it indicates that *Device Auto-Initialization* (DAI) is complete for chip 0.
To enable DAI to complete either wait 10µs or poll the mrr_data Register until it returns 0x0.		
direct_cmd	0x0008FF0A	MODEREG write of 0xFF to mode register 10, to start ZQ calibration for chip 0.
direct_cmd	0x00082201	MODEREG write of 0x22 to mode register 1, to set burst length = 4 for chip 0. Also sets BT, WC, and nWR parameters to default values.
direct_cmd	0x00080102	MODEREG write of 0x01 to mode register 2, to set read latency = 3 and write latency = 1 for chip 0.

If a configured LPDDR2 DMC supports more than one memory chip then repeat the sequence in Table 2-7 but update the chip_addr field to select each additional memory chip.

Duration of ZQ calibration is greater than the duration to complete Mode register write. During this period, **dfi_cke** must not be held LOW. If auto_power_down is enabled, then **dfi_cke** is held LOW after power_down_period, breaking the calibration process. Before proceeding with ZQ calibration, you must disable auto_power_down.

2.4.11 Power-down support

The LPDDR2 DMC provides support for low-power operation in the following ways:

- *Auto power-down*
- *Low-power mode* on page 2-35
- *Deep power-down* on page 2-35
- *Memory clock stopping* on page 2-37
- *Power domains* on page 2-37.

Auto power-down

This feature enables the controller to negate **dfi_cke** if a memory device is idle for a time period of power_dwn_prd **mclk** cycles. This puts the memory device into either active power-down mode or precharge power-down mode, depending on whether the device has any open rows.

You can enable this feature by programming the auto_power_down bit in the memory_cfg Register, see *Memory Configuration Register* on page 3-16. Program the power_dwn_prd field to set the idle time period, in **mclk** cycles.

Figure 2-24 shows the time after completion of a command to a memory chip until the LPDDR2 DMC puts that chip into power-down mode. Power-down affects all the banks of a chip, therefore there might be cases whereby some banks of a chip enters precharge power-down. However, it would normally be expected for at least one bank to enter active power-down.

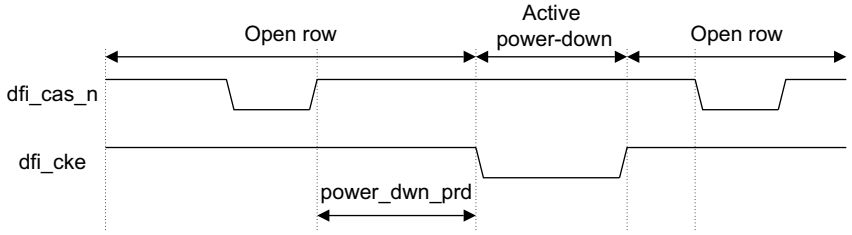


Figure 2-24 Auto power-down

Low-power mode

Low-power mode puts all of the memory devices in to self-refresh mode. You can initiate this mode by either:

Software With the controller in the Ready state then:

1. Issue the Pause command using the memc_cmd Register.
2. Poll the memc_status Register for the Paused state.
3. Issue the Sleep command using the memc_cmd Register.

Hardware Set **csysreq** LOW on the AXI low-power interface.

Note

Do not use low-power mode if any of the memory devices are in deep power-down mode.

To exit low-power mode, you must use the same method that you used to initiate low-power mode. For example, you can only exit low-power mode using the software method if software initiated the low-power mode. Therefore, to exit low-power mode use either:

Software With the controller in the Low_power state then:

1. Issue the `Wakeup` command using the memc_cmd Register.
2. Poll the memc_status Register for the Paused state.
3. Issue the `Go` command and poll for the Ready state.

Hardware Set **csysreq** HIGH on the AXI low-power interface.

Deep power-down

Deep power-down puts one or more of the memory devices in to deep power-down mode, if the LPDDR2 DMC is configured with local **cke**. An LPDDR2 DMC configuration with global **cke** puts all of the SDRAMs into DPD simultaneously.

Note

The system architect must ensure that:

- No data transactions are sent to a memory device that is in deep power-down mode.
- Software tracks which memory devices are in deep power-down mode. The controller does not contain a register that provides the operating mode of an SDRAM.

Deep power-down entry

With the controller in the Ready state, perform the following steps to move one or more memory devices in to deep power-down mode:

1. Issue the `Pause` command using the memc_cmd Register.
2. Poll the memc_status Register for the Paused state.
3. Issue the `Config` command using the memc_cmd Register.
4. Poll the memc_status Register for the Config state.
5. Write to the direct_cmd Register with the `PRECHARGEALL` command. Use the chip_addr field to select the SDRAM to precharge, prior to that device entering deep power-down mode.
6. Write to the direct_cmd Register with the `DPD` command and set the chip_addr field to the value from step 5. The selected SDRAM enters deep power-down mode.

   Note

   If the controller is configured with global **cke** then the controller moves all the SDRAMs to deep power-down mode, irrespective of the chip_addr field value.

7. If the controller is configured with local **cke** then repeat steps 5 and 6 if you require other SDRAMs to enter deep power-down mode.

Deep power-down exit

With the controller in the Ready state, perform the following steps to remove one or more memory devices from deep power-down mode:

1. Issue the `Pause` command using the memc_cmd Register.
2. Poll the memc_status Register for the Paused state.
3. Issue the `Config` command using the memc_cmd Register.
4. Poll the memc_status Register for the Config state.
5. Write to the direct_cmd Register with the `NOP` command. Use the chip_addr field to select the SDRAM to exit from deep power-down mode.

   Note

   If the controller is configured with global **cke** then the controller removes all the SDRAMs from deep power-down mode, irrespective of the chip_addr field value.

6. If the controller is configured with local **cke** then repeat step 5 if you require other SDRAMs to exit from deep power-down mode.

When a memory device exits from deep power-down mode, you must initialize the device prior to it being accessible to the system software.

For examples, see *LPDDR device initialization* on page 2-32 and *LPDDR2-S2 device initialization* on page 2-33.

Memory clock stopping

Memory clock stoppping feature enables the controller to stop the memory clock if the memory devices enter self-refresh mode or deep power-down mode. You can enable this feature by programming the stop_mem_clock bit in the memory_cfg Register, see *Memory Configuration Register* on page 3-16.

If the memory devices enter self-refresh mode then the controller signals to the DFI PHY to disable all of the memory clocks by setting **dfi_dram_clk_disable[MEMORY_CHIPS–1:0]** HIGH.

If a memory device enters deep power-down mode then the controller signals to the DFI PHY to disable the memory clock for that device by setting the appropriate **dfi_dram_clk_disable[MEMORY_CHIPS–1:0]** HIGH.

Power domains

It is possible to implement the LPDDR2 DMC with two power domains:

- APB and AXI, **aclk**
- memory, **mclk**. Figure 2-1 on page 2-2 shows these clock domains.

2.4.12 Power-down usage model

The following sections describe:

- *System states*
- *State transitions* on page 2-39.

System states

Table 2-8 shows the valid system states for both clock domains in the controller and the memory devices.

Table 2-8 Valid system states for FSMs

State	Memory device		LPDDR2 DMC aclk FSM			LPDDR2 DMC mclk FSM		
	V_{DD}	State	V_{DD}	aclk	State	V_{DD}	mclk	State
Start	0	Null	0	-	Null	0	-	Null
1	0	Null	>0	Running	POR	>0	Running	POR
2	0	Null	>0	Running	Reset[a]	>0	Running	Reset[a]
3	0	Null	>0	Running	Config	>0	Running	Pwr_up
4	>0	Accessible	>0	Running	Config	>0	Running	Pwr_up
5	>0	Accessible	>0	Running	Ready	>0	Running	Pwr_up
6	>0	Power-down	>0	Running	Ready	>0	Running	Pwr_down
7	>0	Self-refresh	>0	Running	Low_power	>0	Running	Pwr_sref
8	>0	Self-refresh	>0	Running	Low_power	>0	Stopped	Pwr_sref
9	>0	Self-refresh	>0	Stopped	Low_power	>0	Running	Pwr_sref
10	>0	Self-refresh	>0	Stopped	Low_power	>0	Stopped	Pwr_sref
m1	0 or >0	Deep power-down	>0	Running	Config or Ready	>0	Running	Pwr_dpd
m2	0 or >0	Deep power-down	>0	Running	Config or Ready	>0	Stopped	Pwr_dpd
m3	0 or >0	Deep power-down	>0	Stopped	Config or Ready	>0	Running	Pwr_dpd
m4	0 or >0	Deep power-down	>0	Stopped	Config or Ready	>0	Stopped	Pwr_dpd

a. **aresetn** and **mresetn** are LOW.

The ranking of system power states, from highest power to lowest power, is:

- 5, 6, 7, 9, 8, 10, m1, m3, m2, m4.

States 9, 10, m3, and m4 stop **aclk**. However, this is only possible if an SoC provides logic to enable or disable **aclk**.

Table 2-9 shows a recommended set of power states.

Table 2-9 Recommended power states

System state	Power name
5	Running
6	Auto power-down
8[a]	Self-refresh
m2[b]	Deep power-down

a. Use state 10 if the SoC supports **aclk** stopping.
b. Use state m4 if the SoC supports **aclk** stopping.

State transitions

Figure 2-25 on page 2-40 shows the **aclk** FSM transitions for each of the low-power operating modes that the controller supports. It also shows the **mclk** FSM as it transitions to and from the Pwr_dpd state.

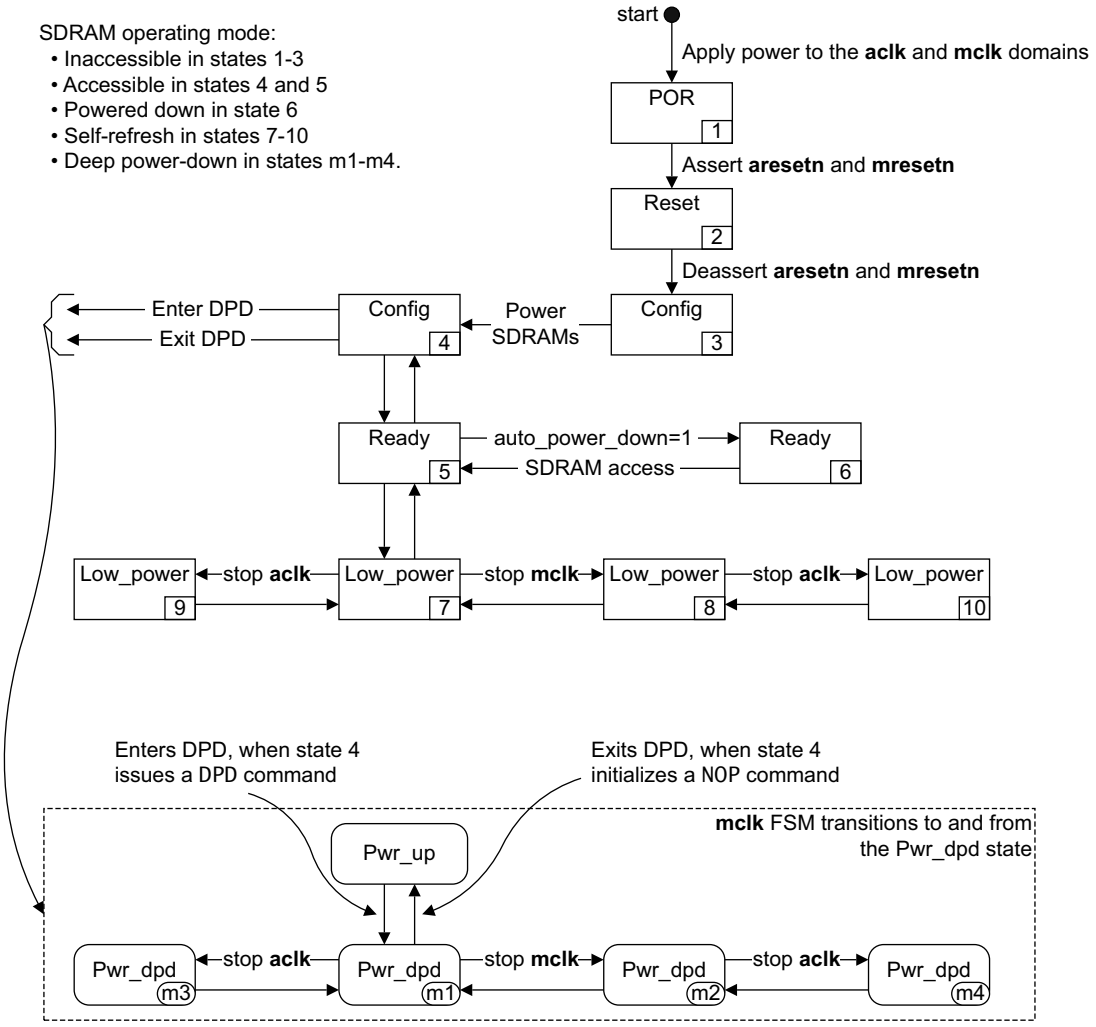


Figure 2-25 aclk FSM and power state transitions

In Figure 2-25, the **aclk** FSM and **mclk** FSM state transitions are:

Start to 1 Apply power to all LPDDR2 DMC power domains and ensure that **aclk** and **mclk** are running.

Arc 1 to 2 Assert reset in the **aclk** domain and the **mclk** domain.

Arc 2 to 3 Deassert reset in the **aclk** domain and the **mclk** domain.

Arc 3 to 4 Apply power to the memory device power domain.

Arc 4 to 5 Initializes the controller and memory devices, see *Initialization* on page 2-29.

Arc 5 to 4 Moves the controller to the Config state so that you can:

- reconfigure the controller
- reconfigure a memory device
- move a memory device to or from deep power-down mode.

To move the controller to the Config state:

1. Write the `Pause` command to the memc_cmd Register.
2. Poll the memc_status Register until the memc_status field returns b10, Paused.
3. Write the `Config` command to the memc_cmd Register.
4. Poll the memc_status Register until the memc_status field returns b00, Config. See *Memory Controller Status Register* on page 3-10 and *Memory Controller Command Register* on page 3-12.

Arc 5 to 6, Active power-down entry or Precharge power-down entry

If the auto_power_down bit is set in the memory_cfg Register then this arc is taken when the memory device has been idle for power_dwn_prd **mclk** cycles. See *Auto power-down* on page 2-34.

Arc 6 to 5, Active power-down exit or Precharge power-down exit

When the controller receives an AXI transfer for a memory device then the corresponding **mclk** FSM moves from the Pwr_down state to the Pwr_up state.

Arc 5 to 7, Low_power entry

Use this arc to move all the SDRAMs to self-refresh mode, by using either hardware or software control. See *Low-power mode* on page 2-35.

Note

When all memory devices are in DPD mode, the **aclk** FSM cannot enter Low_power state. The FSM would continue to remain in Config state

Arc 7 to 5, Low_power exit

Use this arc to remove all the SDRAMs from self-refresh mode, by using either hardware or software control. See *Low-power mode* on page 2-35.

Arc 7 to 8 If the stop_mem_clock bit is set in the memory_cfg Register then this arc is taken when the SDRAM enters self-refresh mode. The controller stops the clock to the SDRAM. See *Memory clock stopping* on page 2-37.

Arc 8 to 7 Either the stop_mem_clock bit is set to 0 or the SDRAM exits self-refresh mode. The controller starts the clock to the SDRAM.

Arc 7 to 9, Arc 8 to 10

The system stops the **aclk** domain clock.

Arc 9 to 7, Arc 10 to 8

The system starts the **aclk** domain clock.

Note

Logic external to the controller is required to start or stop **aclk**.

Deep power-down entry

With the controller in the Config state:

1. Write the `PRECHARGEALL` command to the direct_cmd Register and select the memory device to enter deep power-down mode.
2. Write the `DPD` command to the direct_cmd Register. This puts the selected memory device in to deep power-down mode. See *Deep power-down* on page 2-35.

Deep power-down exit

With the controller in the Config state:

1. Write the `NOP` command to the direct_cmd Register. The selected memory device exits deep power-down mode. See *Deep power-down* on page 2-35.
2. Initialize the memory device, see *Initialization* on page 2-29.

Note

Disable the memory clock stop feature before Deep power-down exit.

Firmware has to send two NOP commands when submitting a request to exit from Deep Power-Down state. The first NOP command is required to exit from Deep Power-Down state and the second NOP command is required before the controller initialize the memory device.

Arc m1 to m2

If the stop_mem_clock bit is set in the memory_cfg Register then this arc is taken when the SDRAM enters deep power-down mode. The controller stops the clock to the SDRAM. See *Memory clock stopping* on page 2-37.

Arc m2 to m1

Either the stop_mem_clock bit is set to 0 or the SDRAM exits deep power-down mode. The controller starts the clock to the SDRAM.

Arc m1 to m3, Arc m2 to m4

The system stops **aclk**.

Arc m3 to m1, Arc m4 to m2

The system starts **aclk**.

Note

Logic external to the controller is required to start or stop **aclk**.

2.5 Constraints and limitations of use

The LPDDR2 DMC supports use with:

- LPDDR2-SDRAM devices but does not support LPDDR2 NVM devices
- LPDDR SDRAM devices but does not support LPDDR NVM devices.

2.5.1 TrustZone Technology Support

The LPDDR2 DMC provides very minimal support related to the security goals as set out in the TrustZone technology. This document outlines the limitations of the controller and the support it requires from other peripherals to meet the security goals of TrustZone technology.

Memory controller in context

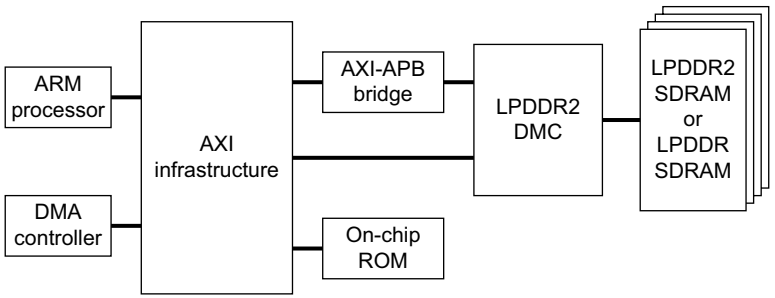


Figure 2-26 LPDDR2 DMC in context

The LPDDR2 DMC has an AXI slave interface and an APB interface. The APB interface has to be placed in a location which is only addressable by the Secure world. The AXI slave interface can be addressed by both the Secure and Non-secure world.

The LPDDR2 DMC does not distinguish between secure read or write data transactions and non-secure access on its AXI or APB interfaces.

There are 3 possible security configurations that need to be considered.

Note

Secure memory and non-secure memory only refer to access control of memory locations as set out in the TrustZone technology.

Non-secure memory

All memory locations addressed through the LPDDR2 DMC are not protected by TrustZone technology and they are accessible to either the Secure or Non-secure worlds. The AXI and APB interfaces of the LPDDR2 DMC are mapped to be accessible by the Secure and Non-secure worlds.

Dedicated secure memory

All memory locations addressed through the LPDDR2 DMC are always protected by TrustZone technology and are accessible only to the Secure world. The AXI and APB regions that connect to the LPDDR2 DMC must be mapped as secure.

Shared secure and non-secure memory

Memory locations addressed through the LPDDR2 DMC contains both secure and non-secure memory regions. An AXI bridging device is required to meet the security goals of TrustZone technology.

APB interface

When any of the regions addressed by an LPDDR2 DMC are made secure on the AXI slave interface, then the control of the mapping of the memories at the LPDDR2 DMC has to be made equally secure by ensuring that only the Secure world can access the APB interface.

AXI slave interface

The LPDDR2 DMC does not distinguish between Secure and Non-secure world access to the AXI slave interface. Memory transactions are not aborted. It does not make the read data as null when a non-secure read request tries to access a secure memory region. Write strobes are not disabled when a non-secure write request tries to access a secure memory region. Therefore the AXI slave interface has to be access controlled to meet the security goals of TrustZone technology.

Memory regions

The LPDDR2 DMC can support a maximum of four chip-selects (memory devices). It supports 32-bits of AXI address bus. Hence it can access up to 4 GBytes of data. This memory range can be divided into multiple memory regions up to a maximum of four. This is defined by individual chip_<n>_cfg registers that are programmable through the APB interface. Each of the individual chip_<n>_cfg registers has an address match field and an address mask field.

Chip configuration Register map

Chip configuration Register can only be read or written in the Config or Low_power states. See *Chip Configuration registers* on page 3-45.

It is possible to define overlapping memory regions. This can result in two different AXI addresses mapping to the same memory region. Therefore overlapping memory regions must not be defined and control of the mapping of the memories at the LPDDR2 DMC must be made secure by ensuring that only the Secure world can access the APB interface.

If the LPDDR2 DMC receives an AXI access that does not map to a chip select then the controller performs the access to chip select 0. If the memory device that connects to chip select 0 contains secure data then it is possible for a master in Non-secure state to access that data. To prevent this security violation from occurring, you must ensure that the interconnect or system that connects to the controller, can only issue accesses that map directly to a chip select.

Side-channel interface (clock, reset, and power)

The programmable voltage and frequency sources inside a system must stay within specification, so the device that controls the voltage and frequency must be secured by TrustZone access control techniques.

Caution

TrustZone security extensions enables a secure software environment. The technology does not protect the processor, LPDDR2 DMC, or other peripherals from hardware attacks and the implementer must take appropriate steps to secure the hardware and protect the trusted code.

2.5.2 Supported memory devices

ARM has tested the LPDDR2 DMC with various LPDDR and LPDDR2 SDRAMs from several memory vendors. For more information, see the *AMBA LPDDR2 Dynamic Memory Controller DMC-342 Release Note*.

Chapter 3
Programmers Model

This chapter describes the LPDDR2 DMC registers and provides information about programming the controller. It contains the following sections:

3.1 About the programmers model

The following information applies to the LPDDR2 DMC registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these location can result in unpredictable behavior.
- Unless otherwise stated in the accompanying text:
  - — do not modify undefined register bits
  - — ignore undefined register bits on reads
  - — all register bits are reset to a logic 0 by a system or power-on reset.
- Access type in *Register summary* on page 3-7 is described as follows:

  RW Read and write.

  RO Read only.

  WO Write only.

3.1.1 Register map

The register map of the LPDDR2 DMC spans a 4KB region, see Figure 3-1.

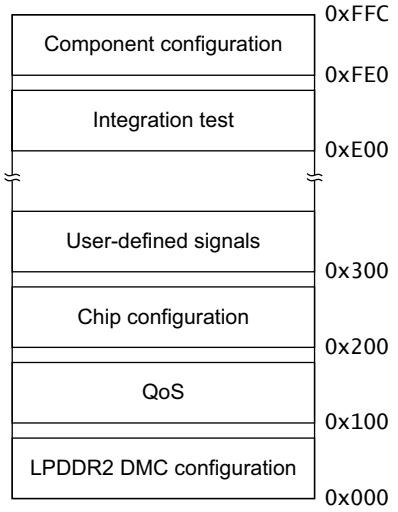


Figure 3-1 Register map

In Figure 3-1 the register map consists of the following main blocks:

- *LPDDR2 DMC configuration* on page 3-3
- *QoS registers* on page 3-5

LPDDR2 DMC configuration

Figure 3-2 on page 3-4 shows the LPDDR2 DMC configuration Register map.

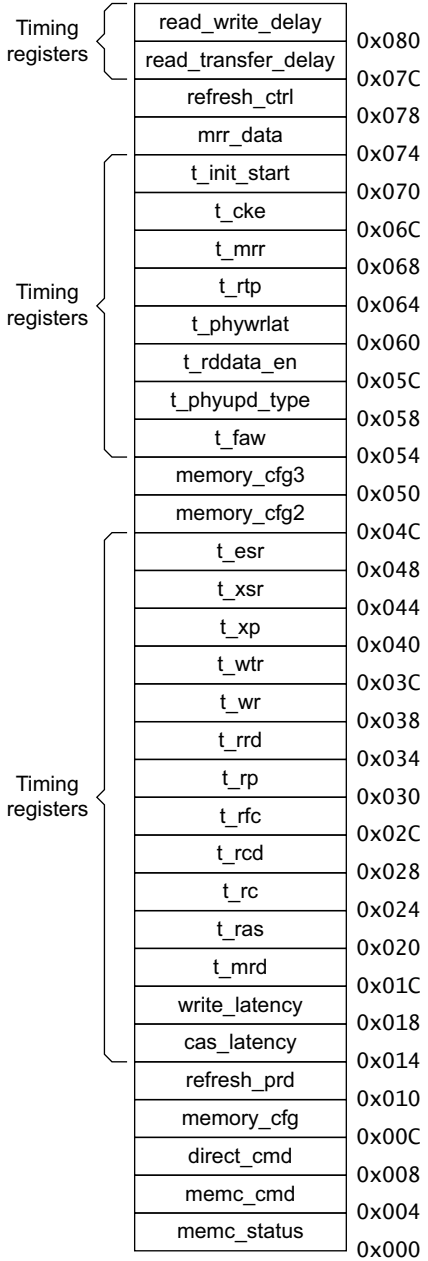


Figure 3-2 Configuration register map

QoS registers

Figure 3-3 shows the QoS Register map.

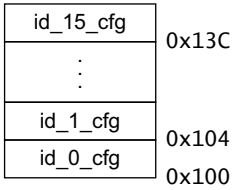


Figure 3-3 QoS register map

Chip configuration

Figure 3-4 shows the chip configuration Register map.

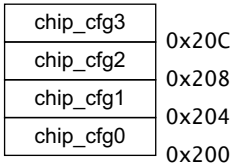


Figure 3-4 Chip configuration register map

User registers

Figure 3-5 shows the memory map for the Feature Control Register and the following user signals:

- **user_config1[]**
- **user_config0[]**
- **user_status[]**.

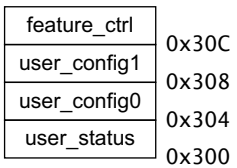


Figure 3-5 User register map

Integration test

Use these registers to verify correct integration of the LPDDR2 DMC in a system, by enabling non-AMBA signals to be set and read. Figure 3-6 shows the integration test Register map.

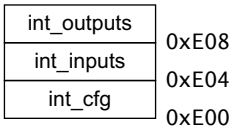


Figure 3-6 Integration test register map

Component configuration

Figure 3-7 shows the Component configuration Register map.

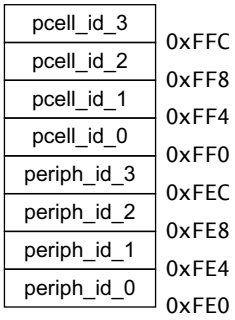


Figure 3-7 Component configuration register map

3.2 Register summary

Table 3-1 shows the LPDDR2 DMC registers in base offset order.

Table 3-1 LPDDR2 DMC register summary

Offset	Name	Type	Reset	Description
0x000	memc_status	RO	-[a]	*Memory Controller Status Register* on page 3-10
0x004	memc_cmd	WO	-	*Memory Controller Command Register* on page 3-12
0x008	direct_cmd	WO	-	*Direct Command Register* on page 3-14
0x00C	memory_cfg	RW	0x00010020	*Memory Configuration Register* on page 3-16
0x010	refresh_prd	RW	0x000003D0	*Refresh Period Register* on page 3-18
0x014	cas_latency	RW	0x00000003	*CAS Latency Register* on page 3-19
0x018	write_latency	RW	0x00000001	*Write Latency Register* on page 3-20
0x01C	t_mrd	RW	0x00000002	*MODEREG to Command Timing Register* on page 3-20
0x020	t_ras	RW	0x0000000E	*ACTIVE to PRECHARGE Timing Register* on page 3-21
0x024	t_rc	RW	0x00000012	*ACTIVE to ACTIVE Timing Register* on page 3-22
0x028	t_rcd	RW	0x00000205	*ACTIVE to Read or Write Timing Register* on page 3-23
0x02C	t_rfc	RW	0x00002023	*AUTO REFRESH to Command Timing Register* on page 3-23
0x030	t_rp	RW	0x00000205	*PRECHARGE to Command Timing Register* on page 3-24
0x034	t_rrd	RW	0x00000004	*ACTIVE to ACTIVE Different Bank Timing Register* on page 3-25
0x038	t_wr	RW	0x00000005	*Write to PRECHARGE Timing Register* on page 3-26
0x03C	t_wtr	RW	0x00000004	*Write to Read Timing Register* on page 3-26
0x040	t_xp	RW	0x00000002	*Exit Power-down Timing Register* on page 3-27
0x044	t_xsr	RW	0x00000027	*Exit Self-refresh Timing Register* on page 3-28
0x048	t_esr	RW	0x00000014	*Self-refresh to Command Timing Register* on page 3-28
0x04C	memory_cfg2	RW	-[a]	*Memory Configuration 2 Register* on page 3-29
0x050	memory_cfg3	RW	0x00000007	*Memory Configuration 3 Register* on page 3-32

Table 3-1 LPDDR2 DMC register summary (continued)

Offset	Name	Type	Reset	Description
0x054	t_faw	RW	0x00001114	*Four Activate Window Timing Register* on page 3-33
0x058	t_phyupd_type	RW	0x00000000	*Update Type Register* on page 3-33
0x05C	t_rddata_en	RW	0x00000003	*Read Data Enable Timing Register* on page 3-35
0x060	t_phywrlat	RW	0x00000000	*Write Data Enable Timing Register* on page 3-35
0x064	t_rtp	RW	0x00000002	*Read to PRECHARGE Timing Register* on page 3-36
0x068	t_mrr	RW	0x00000002	*Read Mode Register* on page 3-37
0x06C	t_cke	RW	0x00000002	*Clock Enable Register* on page 3-38
0x070	t_init_start	RW	0x00000004	*DFI Initialize Register* on page 3-39
0x074	mrr_data	RO	0x00000000	*Mode Read Data Register* on page 3-40
0x078	refresh_ctrl	RW	0x00000005	*Refresh Control Register* on page 3-41
0x07C	read_transfer_delay	RW	0x00000001	*Read Transfer Delay Register* on page 3-42
0x080	read_write_delay	RW	0x00000002	*Read Write Delay Register* on page 3-43
0x084-0x0FC	-	-	-	Reserved, read undefined, write as zero
0x100-0x13C	id_<n>_cfg	RW	0x00000000	*QoS Configuration registers* on page 3-44
0x140-0x1FC	-	-	-	Reserved, read undefined, write as zero
0x200 0x204[b] 0x208[b] 0x20C[b]	chip_cfg0 chip_cfg1 chip_cfg2 chip_cfg3	RW	0x0000FF00	*Chip Configuration registers* on page 3-45
0x210-0x2FC	-	-	-	Reserved, read undefined, write as zero
0x300	user_status	RO	-	*User Status Register* on page 3-46
0x304	user_config0	WO	-	*User Config 0 Register* on page 3-47
0x308	user_config1	WO	-	*User Config 1 Register* on page 3-48
0x30C	feature_ctrl	RW	0x00000000	*Feature Control Register* on page 3-49
0x310-0xDFC	-	-	-	Reserved, read undefined, write as zero
0xE00	int_cfg	RW	0x00000000	*Integration Configuration Register* on page 3-50

Table 3-1 LPDDR2 DMC register summary (continued)

Offset	Name	Type	Reset	Description
0xE04	int_inputs	RO	-[c]	*Integration Inputs Register* on page 3-51
0xE08	int_outputs	WO	-	*Integration Outputs Register* on page 3-52
0xE0C-0xFDC	-	-	-	Reserved, read undefined, write as zero
0xFE0-0xFEC	periph_id_n	RO	0x00_41342[d]	*Peripheral Identification registers* on page 3-53
0xFF0-0xFFC	pcell_id_n	RO	0xB105F00D	*Component Identification registers* on page 3-55

a. Dependent on configuration.
b. The presence of this register depends on the number of chip selects that a configured controller supports. If a controller does not implement the register then reads are undefined, write as zero.
c. Dependent on the state of various input signals, see *Integration Inputs Register* on page 3-51.
d. Dependent on the revision of the LPDDR2 DMC, see *Peripheral Identification Register 2* on page 3-55.

3.3 Register descriptions

This section describes the LPDDR2 DMC registers.

3.3.1 Memory Controller Status Register

The memc_status Register characteristics are:

Purpose Provides information about the configuration and current state of the LPDDR2 DMC.

Usage constraints Not accessible in the Reset or *Power-On Reset* (POR) state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-8 shows the memc_status Register bit assignments.

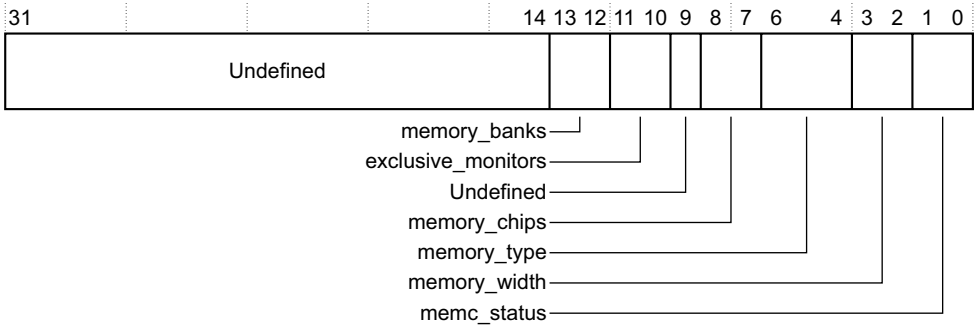


Figure 3-8 memc_status Register bit assignments

Table 3-2 shows the memc_status Register bit assignments.

Table 3-2 memc_status Register bit assignments

Bits	Name	Function
[31:14]	-	Read undefined.
[13:12]	memory_banks	Returns the maximum number of banks per memory chip: b00 = 1 bank, not supported b01 = 2 banks, not supported b10 = 4 banks b11 = 8 banks. Memory Banks per Chip[a] option configures this field.
[11:10]	exclusive_monitors	Returns the number of exclusive access monitor resources implemented in the controller: b00 = 0 monitors b01 = 1 monitor b10 = 2 monitors b11 = 4 monitors. Exclusive Monitors[a] option configures this field.
[9]	-	Undefined, read as zero.
[8:7]	memory_chips	Returns the number of different chip selects that a configured controller supports: b00 = 1 chip b01 = 2 chips b10 = 3 chips b11 = 4 chips. Memory Chips[a] option configures this field.

Table 3-2 memc_status Register bit assignments (continued)

Bits	Name	Function
[6:4]	memory_type	Returns the type of memory devices that a configured controller supports: b000-b110 = reserved b111 = LPDDR, LPDDR2-S2, and LPDDR2-S4.
[3:2]	memory_width	Returns the memory data bus width, MEMWIDTH, between the PHY and the memory devices: b00 = 16-bit b01 = 32-bit b10 = reserved b11 = reserved. Memory Bus Width[a] option configures this field.
[1:0]	memc_status	Returns the state of the memory controller: b00 = Config b01 = Ready b10 = Paused b11 = Low_power.

a. Use AMBA Designer to configure this option.

3.3.2 Memory Controller Command Register

The memc_cmd Register characteristics are:

Purpose Controls the operating state of the LPDDR2 DMC.

Usage constraints Not accessible in the Reset or *Power On Reset* (POR) state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-9 shows the memc_cmd Register bit assignments.

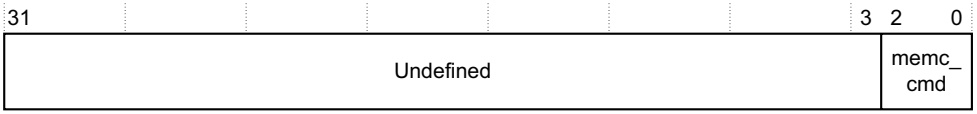


Figure 3-9 memc_cmd Register bit assignments

Table 3-3 shows the memc_cmd Register bit assignments.

Table 3-3 memc_cmd Register bit assignments

Bits	Name	Function
[31:3]	-	Undefined, write as zero.
[2:0]	memc_cmd	Use the following commands to change the state of the LPDDR2 DMC: b000 = Go b001 = Sleep b010 = Wakeup b011 = Pause b100 = Configure b111 = Active_Pause. If the controller receives a command to change state and a previous command to change state has not completed then it holds **pready** LOW until the new command can be carried out. See *aclk domain state diagram* on page 2-21 for more information about the state transitions.

Note

- `Active_Pause` command puts the LPDDR2 DMC into the Paused state without draining the arbiter queue. This enables you to move the controller to the Low_power state, to change configuration settings such as memory frequency or timing register values, without requiring co-ordination between masters in a multi-master system.
- If you use the `Active_Pause` command to put the DDR2 DMC in the Low_power state then you must not remove power from the controller because this results in data loss and violation of the AXI protocol.
- The LPDDR2 DMC does not issue refreshes when in the Config state. Therefore ARM recommends that you make register updates with the controller in Low_power state because this ensures that the memory is put into self-refresh mode, rather than the Config state when the memory contains valid data.

3.3.3 Direct Command Register

The direct_cmd Register characteristics are:

Purpose Initializes and updates the external memory devices by sending the following commands:

- NOP
- PRECHARGEALL
- AUTO REFRESH
- MODEREG
- DPD.

Usage constraints Only accessible in Config state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

The direct_cmd Register therefore enables any initialization sequence that an external memory device might require. The only timing information associated with the direct_cmd Register are the command delays that are programmed in the timing registers. Figure 3-2 on page 3-4 shows the timing registers. Therefore, if an initialization sequence requires additional delays between commands, they must be timed by the master driving the initialization sequence.

Figure 3-10 shows the direct_cmd Register bit assignments.

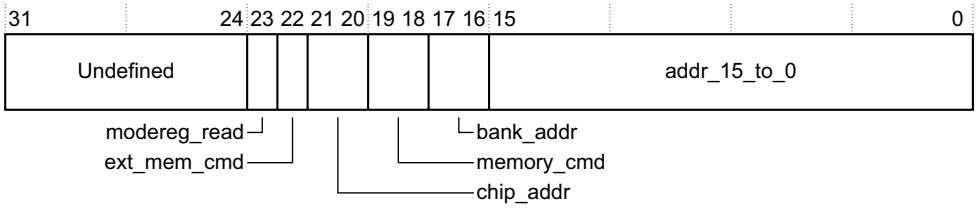


Figure 3-10 direct_cmd Register bit assignments

Table 3-4 shows the direct_cmd Register bit assignments.

Table 3-4 direct_cmd Register bit assignments

Bits	Name	Function
[31:24]	-	Undefined, write as zero.
[23]	modereg_read	When the LPDDR2 DMC performs a MODEREG command to an LPDDR2 device, set this to the access type: 0 = MODEREG write 1 = MODEREG read. The controller ignores this bit when it accesses LPDDR devices.
[22]	ext_mem_cmd	Extended memory command, see Table 3-5.
[21:20]	chip_addr	Bits mapped to external memory chip selects, **dfi_cs_n[MEMORY_CHIPS–1:0]**.
[19:18]	memory_cmd	Selects the command required, see Table 3-5.
[17:16]	bank_addr	Bits mapped to external memory bank address bits, **dfi_bank[1:0]**, when the LPDDR2 DMC issues a MODEREG command.
[15:0]	addr_15_to_0	Bits mapped to external memory address bits, **dfi_address[15:0]**, when the LPDDR2 DMC issues a MODEREG command.

Table 3-5 shows the memory command encoding from the setting of the ext_mem_cmd and memory_cmd bits.

Table 3-5 Memory command encoding

ext_mem_cmd	memory_cmd	Command
0	b00	PRECHARGEALL.
0	b01	AUTO REFRESH.
0	b10	MODEREG or Extended MODEREG access.
0	b11	NOP. A NOP command asserts all chip selects that are set as active_chips when the chip_addr is set to 0. The active_chips field is in the *Memory Configuration Register* on page 3-16. If chip_addr is set to: • 1, only **dfi_cs_n[1]** is asserted. • 2, only **dfi_cs_n[2]** is asserted. • 3, only **dfi_cs_n[3]** is asserted.

Table 3-5 Memory command encoding (continued)

ext_mem_cmd	memory_cmd	Command
1	b00	DPD.
1	b01	-[a]
1	b10	-[a]
1	b11	-[a]

a. Illegal combination that might cause undefined behavior.

Note

For configurations rendered with less than four chip selects, if the controller receives a direct command for a non-existent chip-select then it issues the command to chip select 0.

3.3.4 Memory Configuration Register

The memory_cfg Register characteristics are:

Purpose Controls the operation of the LPDDR2 DMC.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-11 shows the memory_cfg Register bit assignments.

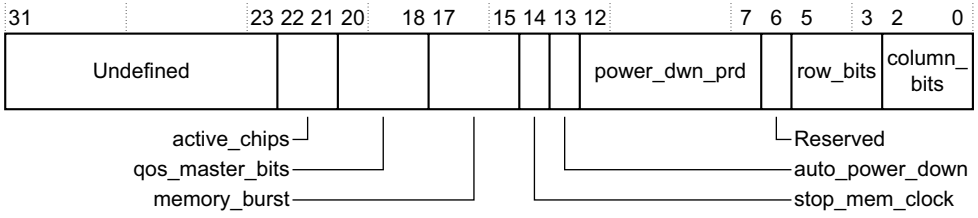


Figure 3-11 memory_cfg Register bit assignments

Table 3-6 shows the memory_cfg Register bit assignments.

Table 3-6 memory_cfg Register bit assignments

Bits	Name	Function
[31:23]	-	Read undefined, write as zero.
[22:21]	active_chips	Enables the LPDDR2 DMC to generate refresh commands for the following number of memory chips: b00 = 1 chip b01 = 2 chips b10 = 3 chips b11 = 4 chips. It is only possible to generate commands up to and including the number of chips in the configuration that the memc_status Register defines, see *Memory Controller Status Register* on page 3-10.
[20:18]	qos_master_bits	Controls which bits of the **arid** bus that the LPDDR2 DMC uses when it selects the QoS value for an AXI read transfer: b000 = **arid[3:0]** b001 = **arid[4:1]** b010 = **arid[5:2]** b011 = **arid[6:3]** b100 = **arid[7:4]** b101 = **arid[8:5]** b110 = **arid[9:6]** b111 = **arid[10:7]**. See *Quality of Service* on page 2-18 for more information.
[17:15]	memory_burst	Controls how many data accesses that the LPDDR2 DMC performs to a memory device, for each `Read` or `Write` command: b010 = burst of 4 b011 = burst of 8 others = reserved. After selecting the burst length you must ensure that device initialization programs the same burst length into the mode register of a memory device. See *LPDDR device initialization* on page 2-32 and *LPDDR2-S2 device initialization* on page 2-33.
[14]	stop_mem_clock	When set to 1, if a memory device enters self-refresh mode or DPD mode then the LPDDR2 DMC sets the corresponding **dfi_dram_clk_disable[MEMORY_CHIPS–1:0]** HIGH. This signals the DFI PHY to stop the clock to the memory device.

Table 3-6 memory_cfg Register bit assignments (continued)

Bits	Name	Function
[13]	auto_power_down	When this is set, the memory interface automatically places the memory devices into power-down state by deasserting **dfi_cke** when the command FIFO has been empty for power_dwn_prd memory clock cycles.
[12:7]	power_dwn_prd	Number of memory clock cycles for auto power-down of the memory devices. You must only change this field when either: • auto_power_down bit is 0 • LPDDR2 DMC is in the Low_power state.
[6]	-	Reserved. Ignored for writes, read as zero.
[5:3]	row_bits	Encodes the number of bits of the AXI address that comprise the row address: b000 = 11 bits b001 = 12 bits b010 = 13 bits b011 = 14 bits b100 = 15 bits b101 = 16 bits b110-b111 = reserved.
[2:0]	column_bits	Encodes the number of bits of the AXI address that comprise the column address: b000 = 8 bits. b001 = 9 bits. b010 = 10 bits. b011 = 11 bits. This means that A0-A9, and A11 are used for column address because A10 is a dedicated AP bit. b100 = 12 bits. b101-b111 = reserved.

3.3.5 Refresh Period Register

The refresh_prd Register characteristics are:

Purpose Controls the memory refresh period in memory clock cycles.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-12 on page 3-19 shows the refresh_prd Register bit assignments.

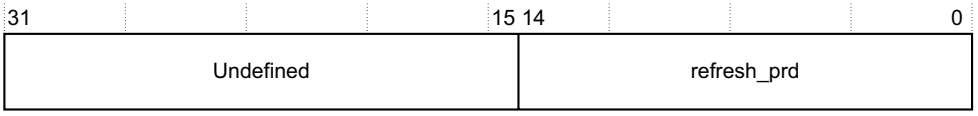


Figure 3-12 refresh_prd Register bit assignments

Table 3-7 shows the refresh_prd Register bit assignments.

Table 3-7 refresh_prd Register bit assignments

Bits	Name	Function
[31:15]	-	Read undefined, write as zero.
[14:0]	refresh_prd	Memory refresh period in **mclk** cycles. Supported values are 63-32767.

3.3.6 CAS Latency Register

The cas_latency Register characteristics are:

Purpose Controls the CAS latency time in memory clock cycles, see Figure 2-23 on page 2-28.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-13 shows the cas_latency Register bit assignments.

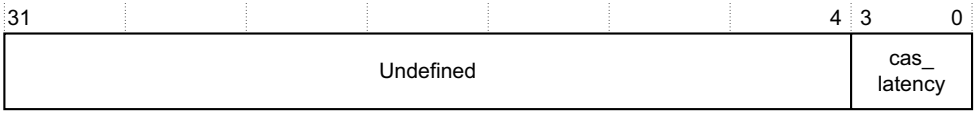


Figure 3-13 cas_latency Register bit assignments

Table 3-8 shows the cas_latency Register bit assignments.

Table 3-8 cas_latency Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero.
[3:0]	cas_latency	CAS latency in **mclk** cycles. Supported values are 3-8.

3.3.7 Write Latency Register

The write_latency Register characteristics are:

Purpose Controls the `Write` to valid DQS strobe delay in memory clock cycles.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-14 shows the write_latency Register bit assignments.

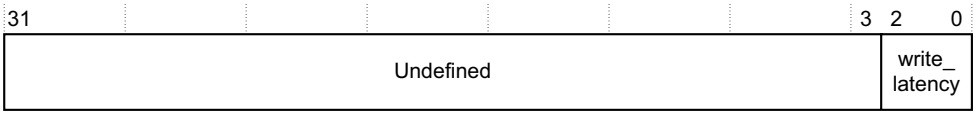


Figure 3-14 write_latency Register bit assignments

Table 3-9 shows the write_latency Register bit assignments.

Table 3-9 write_latency Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	write_latency	Sets the `Write` command to DQS strobe delay in **mclk** cycles. Supported values are 0-7.

Note

When write_latency is set to 1 then the LPDDR2 DMC does not permit the t_phywrlat field, in the t_phywrlat Register, to be set to 1. See *Write Data Enable Timing Register* on page 3-35.

3.3.8 `MODEREG` to Command Timing Register

The t_mrd Register characteristics are:

Purpose Controls the `MODEREG` to command delay in memory clock cycles, see Figure 2-18 on page 2-26.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-15 shows the t_mrd Register bit assignments.

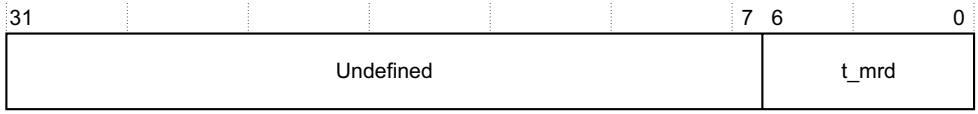


Figure 3-15 t_mrd Register bit assignments

Table 3-10 shows the t_mrd Register bit assignments.

Table 3-10 t_mrd Register bit assignments

Bits	Name	Function
[31:7]	-	Read undefined, write as zero.
[6:0]	t_mrd	Sets t_{MRD}, the time delay, in **mclk** cycles, for the LPDDR2 DMC to issue a command after it issues a MODEREG command. Supported values are 1-127.

3.3.9 ACTIVE to PRECHARGE Timing Register

The t_ras Register characteristics are:

Purpose Controls the RAS to PRECHARGE delay in memory clock cycles, see Figure 2-17 on page 2-26.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-16 shows the t_ras Register bit assignments.

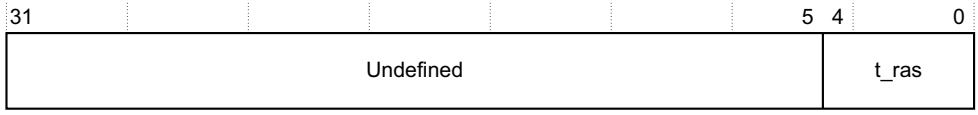


Figure 3-16 t_ras Register bit assignments

Table 3-11 shows the t_ras Register bit assignments.

Table 3-11 t_ras Register bit assignments

Bits	Name	Function
[31:5]	-	Read undefined, write as zero.
[4:0]	t_ras	Sets t_{RAS}, the RAS to PRECHARGE delay in **mclk** cycles. Supported values are 1-31.

3.3.10 ACTIVE to ACTIVE Timing Register

The t_rc Register characteristics are:

Purpose Controls the ACTIVE bank x to ACTIVE bank x delay in memory clock cycles, see Figure 2-14 on page 2-25.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-17 shows the t_rc Register bit assignments.

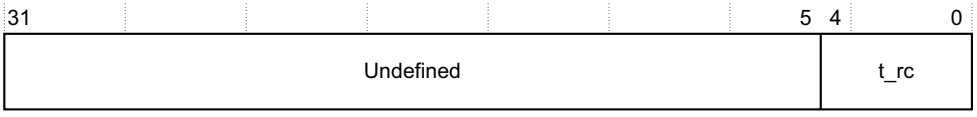


Figure 3-17 t_rc Register bit assignments

Table 3-12 shows the t_rc Register bit assignments.

Table 3-12 t_rc Register bit assignments

Bits	Name	Function
[31:5]	-	Read undefined, write as zero.
[4:0]	t_rc	Sets t_{RC}, the ACTIVE bank x to ACTIVE bank x delay in **mclk** cycles. Supported values are 1-31.

3.3.11 ACTIVE to Read or Write Timing Register

The t_rcd Register characteristics are:

Purpose Controls the RAS to CAS minimum delay in memory clock cycles and controls the delay between an ACTIVE command and another memory command, other than ACTIVE, to the same bank, see Figure 2-12 on page 2-24.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-18 shows the t_rcd Register bit assignments.

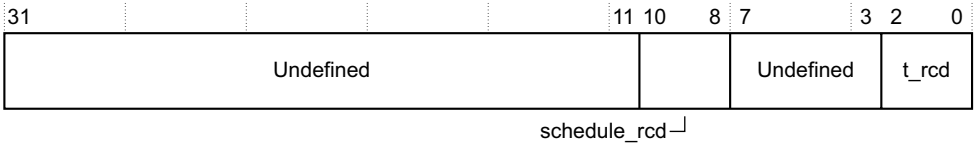


Figure 3-18 t_rcd Register bit assignments

Table 3-13 shows the t_rcd Register bit assignments.

Table 3-13 t_rcd Register bit assignments

Bits	Name	Function
[31:11]	-	Read undefined, write as zero.
[10:8]	schedule_rcd	Sets the RAS to CAS minimum delay in **aclk** clock cycles minus 3. It is used as a scheduler delay and values in the range 0-7 are supported.
[7:4]	-	Read undefined, write as zero.
[3:0]	t_rcd	Sets t_{RCD}, the RAS to CAS minimum delay in **mclk** cycles. Supported values are 0-11.

3.3.12 AUTO REFRESH to Command Timing Register

The t_rfc Register characteristics are:

Purpose Controls the AUTO REFRESH to command delay in memory clock cycles, see Figure 2-16 on page 2-26.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-19 shows the t_rfc Register bit assignments.

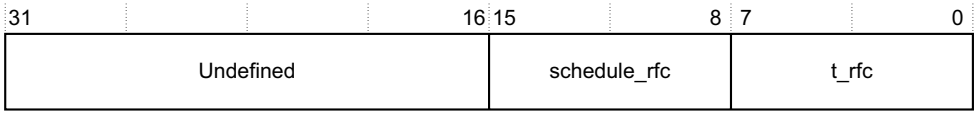


Figure 3-19 t_rfc Register bit assignments

Table 3-14 shows the t_rfc Register bit assignments.

Table 3-14 t_rfc Register bit assignments

Bits	Name	Function
[31:16]	-	Read undefined, write as zero.
[15:8]	schedule_rfc	Sets the AUTO REFRESH to command delay in **aclk** clock cycles minus 3. It is used as a scheduler delay and values of 0-255 are supported.
[7:0]	t_rfc	Sets t_{RFC}, the AUTO REFRESH to command delay in **mclk** cycles. Supported values are 1-255.

3.3.13 PRECHARGE to Command Timing Register

The t_rp Register characteristics are:

Purpose Controls the PRECHARGE to RAS delay in memory clock cycles, see Figure 2-16 on page 2-26 and Figure 2-17 on page 2-26.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-20 shows the t_rp Register bit assignments.

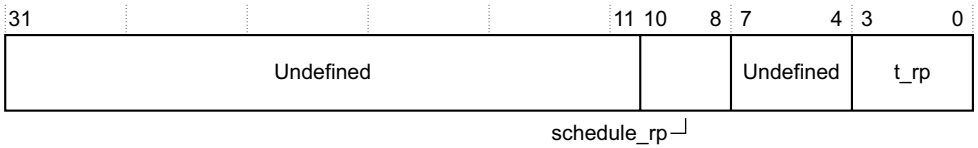


Figure 3-20 t_rp Register bit assignments

Table 3-15 shows the t_rp Register bit assignments.

Table 3-15 t_rp Register bit assignments

Bits	Name	Function
[31:11]	-	Read undefined, write as zero.
[10:8]	schedule_rp	Sets the PRECHARGE to RAS delay in **aclk** clock cycles minus 3. It is used as a scheduler delay and values in the range 0-8 are supported.
[7:4]	-	Read undefined, write as zero.
[3:0]	t_rp	Sets t_{RP}, the PRECHARGE to RAS delay in **mclk** cycles. Supported values are 1-11.

3.3.14 ACTIVE to ACTIVE Different Bank Timing Register

The t_rrd Register characteristics are:

Purpose Controls the ACTIVE bank x to ACTIVE bank y delay in memory clock cycles, see Figure 2-15 on page 2-25.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-21 shows the t_rrd Register bit assignments.

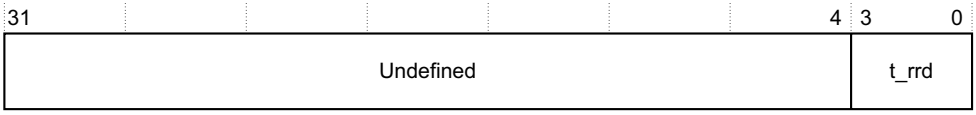


Figure 3-21 t_rrd Register bit assignments

Table 3-16 shows the t_rrd Register bit assignments.

Table 3-16 t_rrd Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero.
[3:0]	t_rrd	Sets t_{RRD}, the ACTIVE bank x to ACTIVE bank y delay in **mclk** cycles. Supported values are 1-15.

3.3.15 Write to PRECHARGE Timing Register

The t_wr Register characteristics are:

Purpose Controls the Write to PRECHARGE delay in memory clock cycles, see Figure 2-22 on page 2-28.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-22 shows the t_wr Register bit assignments.

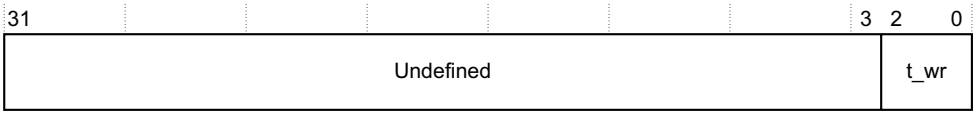


Figure 3-22 t_wr Register bit assignments

Table 3-17 shows the t_wr Register bit assignments.

Table 3-17 t_wr Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	t_wr	Sets t_{WR}, the Write to PRECHARGE delay in **mclk** cycles. Supported values are 1-7.

3.3.16 Write to Read Timing Register

The t_wtr Register characteristics are:

Purpose Controls the Write to Read delay in memory clock cycles, see Figure 2-21 on page 2-28.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-23 on page 3-27 shows the t_wtr Register bit assignments.

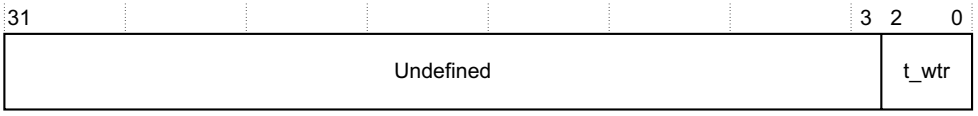


Figure 3-23 t_wtr Register bit assignments

Table 3-18 shows the t_wtr Register bit assignments.

Table 3-18 t_wtr Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	t_wtr	Sets t_{WTR}, the `Write` to `Read` command delay in **mclk** cycles. Supported values are 1-7.

3.3.17 Exit Power-down Timing Register

The t_xp Register characteristics are:

Purpose Controls the exit power-down to command delay in memory clock cycles, see Figure 2-20 on page 2-27.

Usage constraints Only accessible in Config or Low_power state.

You must only write when either:

- auto_power_down bit is 0, see Table 3-6 on page 3-17
- LPDDR2 DMC is in the Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-24 shows the t_xp Register bit assignments.

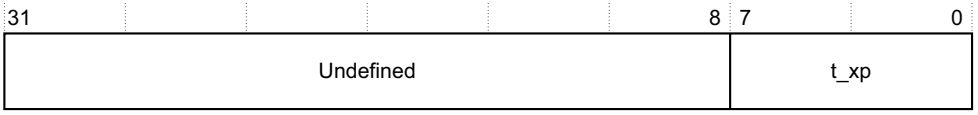


Figure 3-24 t_xp Register bit assignments

Table 3-19 shows the t_xp Register bit assignments.

Table 3-19 t_xp Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:0]	t_xp	Sets t_{XP}, the exit power-down to command delay in **mclk** cycles. Supported values are 1-255.

3.3.18 Exit Self-refresh Timing Register

The t_xsr Register characteristics are:

Purpose Controls the exit self-refresh to command delay in memory clock cycles, see Figure 2-19 on page 2-27.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-25 shows the t_xsr Register bit assignments.

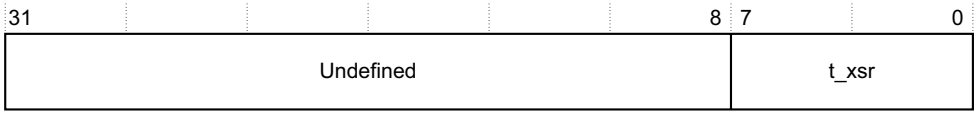


Figure 3-25 t_xsr Register bit assignments

Table 3-20 shows the t_xsr Register bit assignments.

Table 3-20 t_xsr Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:0]	t_xsr	Sets t_{XSR}, the exit self-refresh to command delay in **mclk** cycles. Supported values are 1-255.

3.3.19 Self-refresh to Command Timing Register

The t_esr Register characteristics are:

Purpose Controls the self-refresh to command delay in memory clock cycles, see Figure 2-19 on page 2-27.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-26 shows the t_esr Register bit assignments.

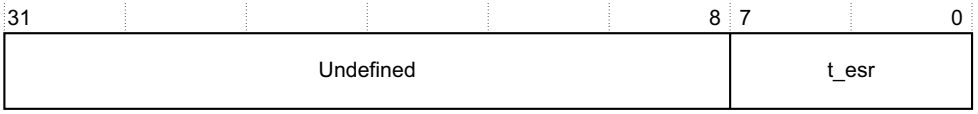


Figure 3-26 t_esr Register bit assignments

Table 3-21 shows the t_esr Register bit assignments.

Table 3-21 t_esr Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:0]	t_esr	Sets the self-refresh to command delay in **mclk** cycles. Supported values are 1-255.

3.3.20 Memory Configuration 2 Register

The memory_cfg2 Register characteristics are:

Purpose Controls the operation of the LPDDR2 DMC and enables you to override the configuration set by some of the tie-off signals, see *Tie-offs* on page A-3.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-27 on page 3-30 shows the memory_cfg2 Register bit assignments.

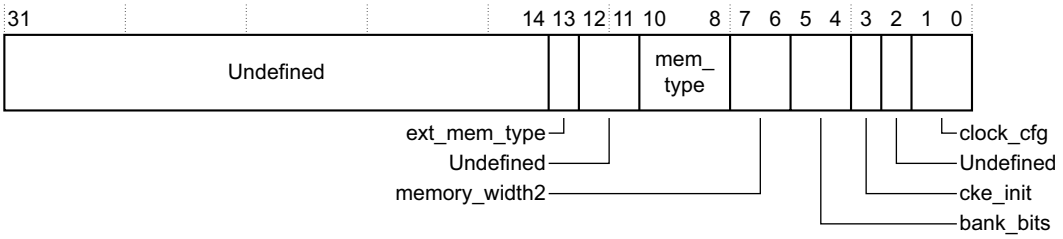


Figure 3-27 memory_cfg2 Register bit assignments

Table 3-22 shows the memory_cfg2 Register bit assignments.

Table 3-22 memory_cfg2 Register bit assignments

Bits	Name	Function
[31:14]	-	Read undefined, write as zero.
[13]	ext_mem_type	Extended memory type, see Table 3-23 on page 3-31.
[12:11]	-	Read undefined, write as zero.
[10:8]	mem_type	Sets the type of memory device that the controller supports, see Table 3-23 on page 3-31.
[7:6]	memory_width2	Controls if the LPDDR2 DMC uses the entire configured memory data bus width or half of the data bus. Depending on the configured MEMWIDTH value, the following bit settings are possible: • b00 = memory device uses a 16-bit data bus. You can select this setting when MEMWIDTH=16 or MEMWIDTH=32. The controller uses the entire data bus when MEMWIDTH=16 and half of the data bus when MEMWIDTH=32. **Note** Do not use this setting if the controller uses an AXI data width of 128 bits. For more information about supported memory widths, see *Features* on page 1-4. • b01 = memory device uses a 32-bit data bus.You can select this setting when MEMWIDTH=32 or MEMWIDTH=64. The controller uses the entire data bus when MEMWIDTH=32 and half of the data bus when MEMWIDTH=64. • b10 = memory device uses a 64-bit data bus.You can select this setting when MEMWIDTH=64. The controller uses the entire data bus. • b11 = reserved. You can determine the configured data bus width, MEMWIDTH, by accessing the memory_width field in the memc_status Register, see *Memory Controller Status Register* on page 3-10. The default value is set by the state of **memory_width[1:0]**, when **aresetn** goes HIGH.

Table 3-22 memory_cfg2 Register bit assignments (continued)

Bits	Name	Function
[5:4]	bank_bits	Controls how many bits of the AXI address are allocated for addressing the banks in a memory device. The options are: b00 = reserved b01 = reserved b10 = 2 bits. Use this when a memory device contains a maximum of 4 banks b11 = 3 bits. Use this when a memory device contains 8 banks and the controller configuration supports 8 banks. You can determine the configured number of supported memory banks by accessing the memory_banks field in the memc_status Register, see *Memory Controller Status Register* on page 3-10. The default value is set by the state of **bank_bits[1:0]**, when **aresetn** goes HIGH.
[3]	cke_init	When **mresetn** is deasserted, the controller uses this bit to set the initial state of **dfi_cke**: 0 = **dfi_cke** is LOW 1 = **dfi_cke** is HIGH. The default value is set by the state of **cke_init**, when **aresetn** goes HIGH.
[2]	-	Read undefined, write as zero.
[1:0]	clock_cfg	Encodes the clocking scheme: b00 = **aclk** and **mclk** are asynchronous b01 = **aclk** and **mclk** are synchronous b10-b11 = reserved. The default value of bit [0] is set by the state of **sync**, when **aresetn** goes HIGH.

Table 3-23 shows the supported memory devices that can be set using the ext_mem_type and mem_type bits.

Table 3-23 Supported memory devices

ext_mem_type	mem_type	Memory device
0	b000-b011	-
0	b010	LPDDR SDRAM. This is the default.
0	b011-b110	-

Table 3-23 Supported memory devices (continued)

ext_mem_type	mem_type	Memory device
0	b111	LPDDR2-S2 SDRAM
1	b000	LPDDR2-S4 SDRAM
1	b001-b111	-

3.3.21 Memory Configuration 3 Register

The memory_cfg3 Register characteristics are:

Purpose Controls the refresh timeout setting when the burst-pause refresh mode is disabled.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-28 shows the memory_cfg3 Register bit assignments.

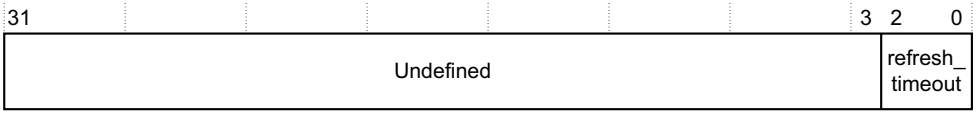


Figure 3-28 memory_cfg3 Register bit assignments

Table 3-24 shows the memory_cfg3 Register bit assignments.

Table 3-24 memory_cfg3 Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	refresh_timeout	When burst-pause refresh is disabled, this field sets the number of acceptable outstanding refreshes for a memory device, before the LPDDR2 DMC generates a refresh timeout. If this occurs, the arbiter entry moves to the highest priority in the queue, see *Arbitration algorithm* on page 2-20. Supported values are 1-7. Use the refresh_control Register to set the burst-pause refresh mode, see *Refresh Control Register* on page 3-41.

3.3.22 Four Activate Window Timing Register

The t_faw Register characteristics are:

Purpose Controls the four bank activate time in memory clock cycles, see Figure 2-13 on page 2-25.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-29 shows the t_faw Register bit assignments.

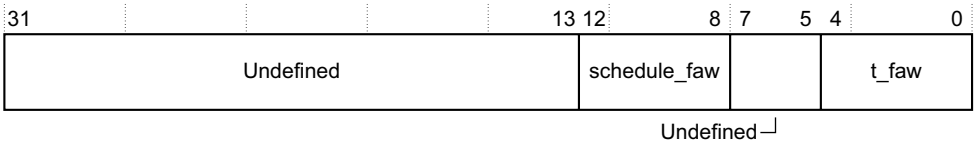


Figure 3-29 t_faw Register bit assignments

Table 3-25 shows the t_faw Register bit assignments.

Table 3-25 t_faw Register bit assignments

Bits	Name	Function
[31:13]	-	Read undefined, write as zero.
[12:8]	schedule_faw	t_faw in **aclk** clock cycles, minus 3. Used as a scheduler delay. Supported values are 0-31.
[7:5]	-	Read undefined, write as zero.
[4:0]	t_faw	Sets t_{FAW}, the four bank activate period in **mclk** cycles. Supported values are 1-31.

3.3.23 Update Type Register

The t_phyupd_type Register characteristics are:

Purpose Controls how the LPDDR2 DMC responds when it receives any of the four possible update type requests from a PHY device.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-30 shows the t_phyupd_type Register bit assignments.

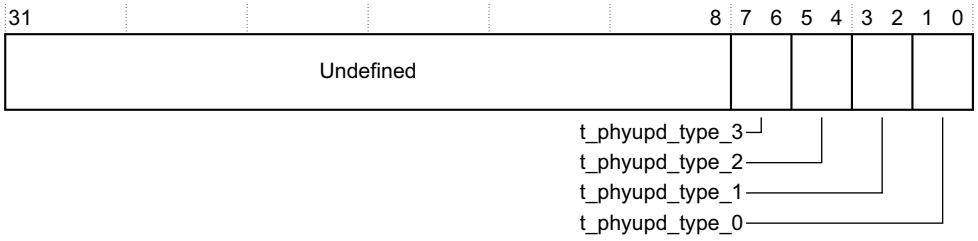


Figure 3-30 t_phyupd_type Register bit assignments

Table 3-26 shows the update_type Register bit assignments.

Table 3-26 t_phyupd_type Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined, write as zero.
[7:6]	t_phyupd_type_3	Controls how the LPDDR2 DMC responds to a DFI update request type 3, that is, when a PHY sets **dfi_phyupd_type[1:0]** to b11: b00 = Put the memory devices in self-refresh mode then stall the DFI b01 = Stall the DFI b10-b11 = Reserved.
[5:4]	t_phyupd_type_2	Controls how the LPDDR2 DMC responds to a DFI update request type 2, that is, when a PHY sets **dfi_phyupd_type[1:0]** to b10: b00 = Put the memory devices in self-refresh mode then stall the DFI b01 = Stall the DFI b10-b11 = Reserved.
[3:2]	t_phyupd_type_1	Controls how the LPDDR2 DMC responds to a DFI update request type 1, that is, when a PHY sets **dfi_phyupd_type[1:0]** to b01: b00 = Put the memory devices in self-refresh mode then stall the DFI b01 = Stall the DFI b10-b11 = Reserved.
[1:0]	t_phyupd_type_0	Controls how the LPDDR2 DMC responds to a DFI update request type 0, that is, when a PHY sets **dfi_phyupd_type[1:0]** to b00: b00 = Put the memory devices in self-refresh mode then stall the DFI b01 = Stall the DFI b10-b11 = Reserved.

3.3.24 Read Data Enable Timing Register

The t_rddata_en Register characteristics are:

Purpose Controls the t_{rddata_en} timing parameter for a DFI pad interface, see Figure 2-23 on page 2-28.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-31 shows the t_rddata_en Register bit assignments.

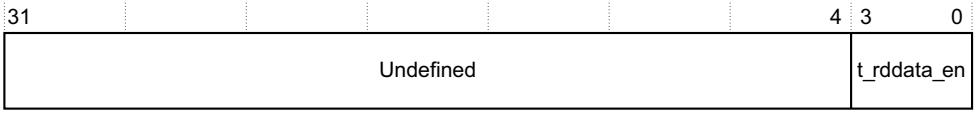


Figure 3-31 t_rddata_en Register bit assignments

Table 3-27 shows the t_rddata_en Register bit assignments.

Table 3-27 t_rddata_en Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero.
[3:0]	t_rddata_en	After a LPDDR2 DMC issues a `Read` command from the DFI pad interface then this field sets the number of **mclk** cycles before **dfi_rddata_en[MEMORY_BYTES–1:0]** goes HIGH. The valid values for this field depend on the cas_latency setting and the limits are: **Minimum** cas_latency−2. **Maximum** cas_latency+5. For example, if cas_latency = 3 then set this field to any value between b0001 and b1000. The cas_latency value is set using the *CAS Latency Register* on page 3-19.

3.3.25 Write Data Enable Timing Register

The t_phywrlat Register characteristics are:

Purpose Controls the t_{phy_wrlat} timing parameter for a DFI pad interface to be either:

- write_latency−1
- write_latency−2.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-32 shows the t_phywrlat Register bit assignments.

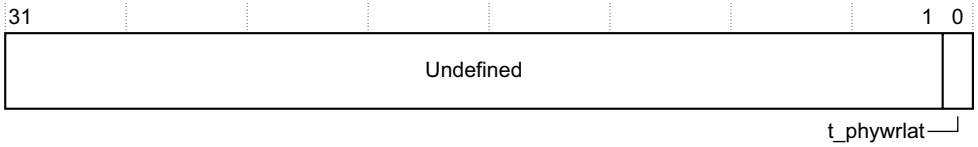


Figure 3-32 t_phywrlat Register bit assignments

Table 3-28 shows the t_phywrlat Register bit assignments.

Table 3-28 t_phywrlat Register bit assignments

Bits	Name	Function
[31:1]	-	Read undefined, write as zero.
[0]	t_phywrlat	Controls the timing relationship between t_{phy_wrlat} and write_latency to be either: 0 = t_{phy_wrlat} is write_latency−1. This is the default. 1 = t_{phy_wrlat} is write_latency−2. The write_latency value is set using the *Write Latency Register* on page 3-20.

3.3.26 Read to PRECHARGE Timing Register

The t_rtp Register characteristics are:

Purpose For LPDDR2 memory devices it controls the Read to PRECHARGE delay, in memory clock cycles.

Note

For LPDDR SDRAM, that is mem_type = b010, then this register has no effect on the operation of the controller. The mem_type field is in the *Memory Configuration 2 Register* on page 3-29.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-33 shows the t_rtp Register bit assignments.

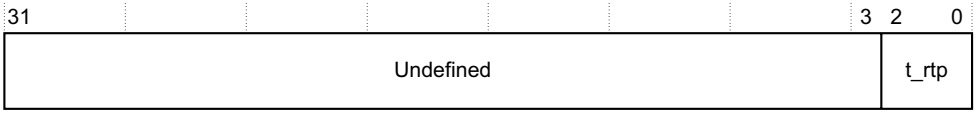


Figure 3-33 t_rtp Register bit assignments

Table 3-29 shows the t_rtp Register bit assignments.

Table 3-29 t_rtp Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	t_rtp	Sets t_{RTP}, the `Read` to `PRECHARGE` delay in **mclk** cycles. Supported values are 0-7.

3.3.27 Read Mode Register

The t_mrr Register characteristics are:

Purpose For LPDDR2 memory devices it controls the duration of the mode register command period, t_{MRR}.

Note

For LPDDR SDRAM, that is mem_type=b010, then this register has no effect on the operation of the controller. The mem_type field is in the *Memory Configuration 2 Register* on page 3-29.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-34 shows the t_mrr Register bit assignments.

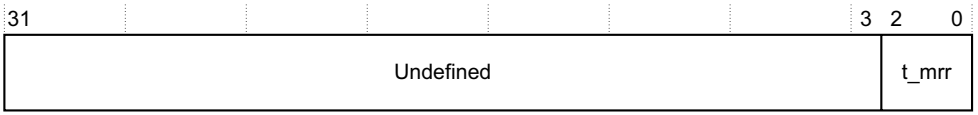


Figure 3-34 t_mrr Register bit assignments

Table 3-30 shows the t_mrr Register bit assignments.

Table 3-30 t_mrr Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	t_mrr	Sets t_{MRR}, the duration of the mode register command period, in **mclk** cycles. Supported values are 0-7.

3.3.28 Clock Enable Register

The t_cke Register characteristics are:

Purpose For LPDDR2 memory devices it controls the minimum number of clock cycles that **dfi_cke** is held LOW. See Figure 2-20 on page 2-27.

Note

For LPDDR SDRAM, that is mem_type=b010, then this register has no effect on the operation of the controller. The mem_type field is in the *Memory Configuration 2 Register* on page 3-29.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-35 shows the t_cke Register bit assignments.

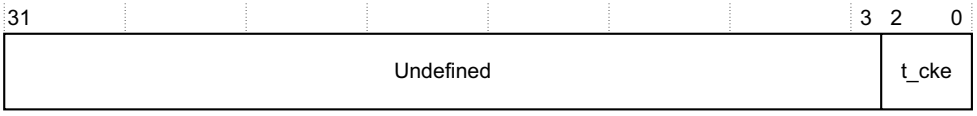


Figure 3-35 t_cke Register bit assignments

Table 3-31 shows the t_cke Register bit assignments.

Table 3-31 t_cke Register bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2:0]	t_cke	Sets t_{CKE}, the number of **mclk** cycles that **dfi_cke** is LOW. Supported values are 0-7.

3.3.29 DFI Initialize Register

The t_init_start Register characteristics are:

Purpose Controls the minimum number of cycles that the LPDDR2 DMC asserts **dfi_init_start**.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-36 shows the t_init_start Register bit assignments.

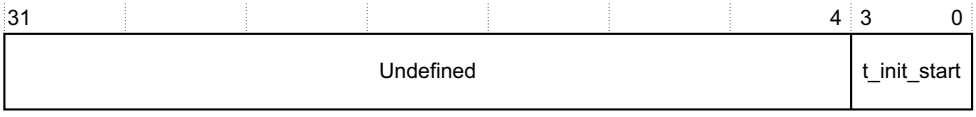


Figure 3-36 t_init_start Register bit assignments

Table 3-32 shows the t_init_start Register bit assignments.

Table 3-32 t_init_start Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero.
[3:0]	t_init_start	Minimum number cycles that the controller asserts **dfi_init_start** when it initializes the DFI PHY. Supported values are 0-15.

3.3.30 Mode Read Data Register

The mrr_data Register characteristics are:

Purpose For LPDDR2 memory devices it returns the read data from a mode register.

Note

For LPDDR SDRAM, that is mem_type=b010, then this register has no effect on the operation of the controller. The mem_type field is in the *Memory Configuration 2 Register* on page 3-29.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-37 shows the mrr_data Register bit assignments.

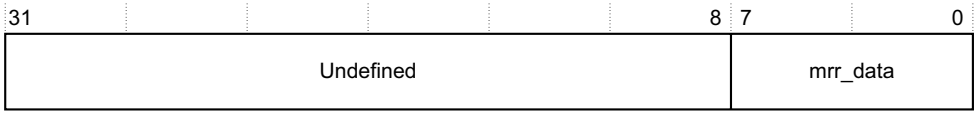


Figure 3-37 mrr_data Register bit assignments

Table 3-33 shows the mrr_data Register bit assignments.

Table 3-33 mrr_data Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined.
[7:0]	mrr_data	Returns the read data from a mode register in an LPDDR2 memory device. The controller uses the data programmed in the direct_cmd Register to select which memory device and mode register to read. Therefore, prior to reading this register you must program the following fields in the direct_cmd Register: **addr_15_to_0** Program bits [7:0] with the mode register number. For example to read MR32, use `0x0020`. **bank_addr** b00. **memory_cmd** b10. Sets a `MODEREG` command if ext_mem_cmd bit is 0. **chip_addr** Program with a value that selects the memory device you require. **ext_mem_cmd** 0. **modereg_read** 1. Sets access type = read.

3.3.31 Refresh Control Register

The refresh_ctrl Register characteristics are:

Purpose For LPDDR2 memory devices it controls the refresh operating modes and refresh timeout.

Note

For LPDDR SDRAM, that is mem_type = b010, then this register has no effect on the operation of the controller. The mem_type field is in the *Memory Configuration 2 Register* on page 3-29.

Usage constraints Only accessible in Low_power state for writes and accessible in Config or Low_power state for reads.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-38 on page 3-42 shows the refresh_ctrl Register bit assignments.

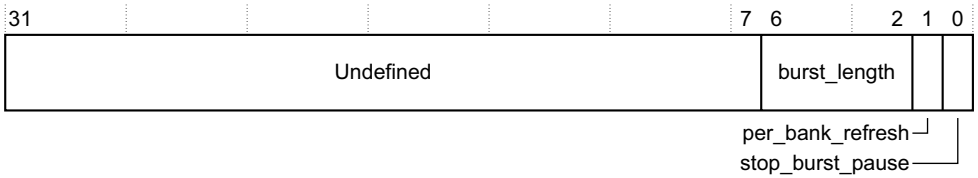


Figure 3-38 refresh_ctrl Register bit assignments

Table 3-34 shows the refresh_ctrl Register bit assignments.

Table 3-34 refresh_ctrl Register bit assignments

Bits	Name	Function
[31:7]	-	Read undefined, write as zero.
[6:2]	burst_length	When burst-pause refresh is enabled, this field sets the number of acceptable outstanding refreshes for an LPDDR2 memory device, before the LPDDR2 DMC generates a refresh timeout. If this occurs, the arbiter entry moves to the highest priority in the queue, see *Arbitration algorithm* on page 2-20. Supported values are 1-16.
[1]	per_bank_refresh	Sets the per-bank refresh mode: 0 = per-bank refresh disabled 1 = per-bank refresh enabled.
[0]	stop_burst_pause	Sets the refresh mode: 0 = burst-pause enabled 1 = burst-pause disabled.

3.3.32 Read Transfer Delay Register

The read_transfer_delay Register characteristics are:

Purpose Controls the number of idle cycles between back-to-back reads to different memory devices.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-39 on page 3-43 shows the read_transfer_delay Register bit assignments.

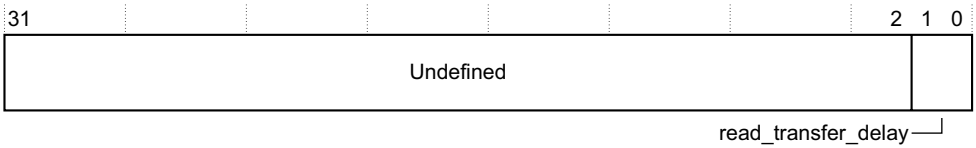


Figure 3-39 read_transfer_delay Register bit assignments

Table 3-35 shows the read_transfer_delay Register bit assignments.

Table 3-35 read_transfer_delay Register bit assignments

Bits	Name	Function
[31:2]	-	Read undefined, write as zero.
[1:0]	read_transfer_delay	When the controller performs back-to-back reads to different memory devices then this field controls the number of idle **mclk** cycles that the controller inserts between the reads: b00 = reserved b01 = one idle cycle b10 = two idle cycles, this is the default b11 = reserved. **Note** • The controller inserts idle cycles to prevent bus contention from occurring, otherwise the second memory device can start driving the DQS bus prior to the first memory device releasing control of the DQS bus. • One idle cycle is sufficient for most memory types. However, for some of the recently introduced LPDDR devices, for example LPDDR-400, then two idle cycles are usually required.

3.3.33 Read Write Delay Register

The read_write_delay Register characteristics are:

Purpose Controls the number of additional cycles that are inserted between back-to-back read and write to the same memory device.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC. It is not applicable to LPDDR memory devices.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-40 on page 3-44 shows the read_write_delay Register bit assignments.

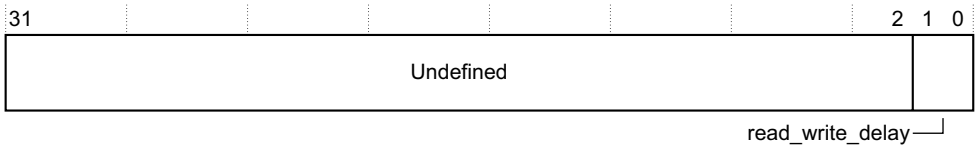


Figure 3-40 read_write_delay Register bit assignments

Table 3-36 shows the read_write_delay Register bit assignments.

Table 3-36 read_write_delay Register bit assignments

Bits	Name	Function
[31:2]	-	Read undefined, write as zero.
[1:0]	read_write_delay	Controls the number of additional cycles to be inserted between back-to-back read and write transactions to the same memory device. These memory devices could be either LPDDR2-S2 or LPDDR2-S4. This is not applicable to LPDDR memory devices. b00 = Reserved b01 = Reserved b10 = Introduces a delay of 2 cycles b11 = Introduces a delay of 3 cycles.

3.3.34 QoS Configuration registers

The id_<n>_cfg Register characteristics are:

Purpose Sets the parameters for QoS <n>. Where <n> is a value from 0 to 15 and is the result of applying the 4-bit QoS mask to the **arid[]** bus.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-41 on page 3-45 shows the id_<n>_cfg Register bit assignments.

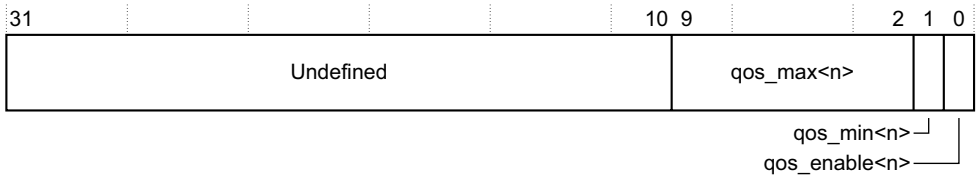


Figure 3-41 id_<n>_cfg registers bit assignments

Table 3-37 shows the id_<n>_cfg Register bit assignments.

Table 3-37 id_<n>_cfg registers bit assignments

Bits	Name	Function
[31:10]	-	Read undefined, write as zero.
[9:2]	qos_max<n>	Sets the maximum QoS value. Supported values are 0-255.
[1]	qos_min<n>	Enables the minimum QoS functionality: 0 = disabled, so the arbiter entry uses the qos_max<n> value. 1 = enabled. the arbiter entry uses minimum latency.
[0]	qos_enable<n>	When set, the LPDDR2 DMC can apply QoS to a read transfer if the masking of the **arid[]** bits with the programmed QoS mask produces a value of <n>. For more information, see *Quality of Service* on page 2-18.

3.3.35 Chip Configuration registers

The chip_cfg<n> Register characteristics are:

Purpose Each register sets the:
- address decode for chip select <n>
- bank, row, column organization of the memory device that connects to chip select <n>.

Usage constraints
- Only accessible in Config or Low_power state.
- The number of registers implemented is equal to the number of chip selects that a configured controller supports.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-42 on page 3-46 shows the chip_cfg<n> Register bit assignments.

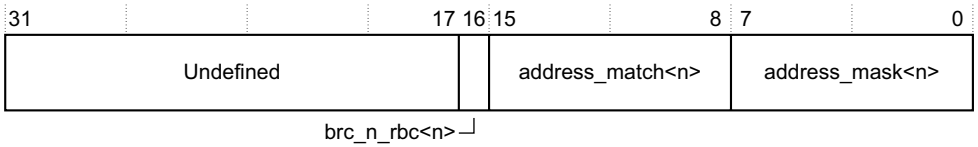


Figure 3-42 chip_cfg<n> Registers bit assignments

Table 3-38 shows the chip_cfg<n> Register bit assignments.

Table 3-38 chip_cfg<n> registers bit assignments

Bits	Name	Function
[31:17]	-	Read undefined, write as zero.
[16]	brc_n_rbc<n>	Selects the memory organization as decoded from the AXI address: 0 = *Row, Bank, Column* (RBC) organization 1 = *Bank, Row, Column* (BRC) organization.
[15:8]	address_match<n>	The controller applies the address_mask<n> field to the AXI address bits, [31:24], that is **awaddr[31:24]** or **araddr[31:24]**. The controller compares the result against this field and if a match occurs then it selects memory device <n>. See *Formatting from AXI address channels* on page 2-12.
[7:0]	address_mask<n>	Controls which AXI address bits, [31:24], the LPDDR2 DMC compares when it receives an AXI transfer: **Bit [*x*] = 0** The controller excludes AXI address bit [24+*x*] from the comparison. **Bit [*x*] = 1** The controller includes AXI address bit [24+*x*] in the comparison.

Note

If a configured controller supports two or more memory devices then you must take care to ensure that for all AXI addresses, you program the various address_match and address_mask fields so that the controller can only assert a single memory chip select. Otherwise, Unpredictable behavior might occur.

3.3.36 User Status Register

The user_status Register characteristics are:

Purpose Provides the status of the **user_status** inputs.

Usage constraints No usage constraints.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-43 shows the user_status Register bit assignments.

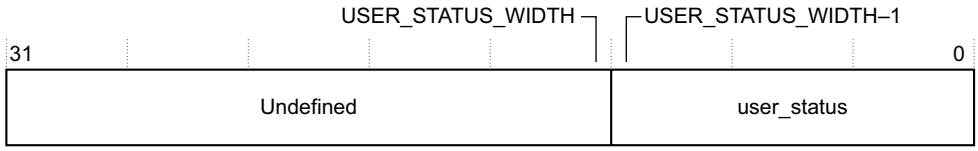


Figure 3-43 user_status Register bit assignments

Table 3-39 shows the user_status Register bit assignments.

Table 3-39 user_status Register bit assignments

Bits	Name	Function
[31:USER_STATUS_WIDTH[a]]	-	Read undefined.
[USER_STATUS_WIDTH–1:0]	user_status	The state of the **user_status[USER_STATUS_WIDTH–1:0]** inputs.

a. If USER_STATUS_WIDTH is configured as 32 then none of the bits in this register are undefined.

3.3.37 User Config 0 Register

The user_config0 Register characteristics are:

Purpose Controls the state of the **user_config0** outputs.

Usage constraints No usage constraints.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-44 shows the user_config0 Register bit assignments.

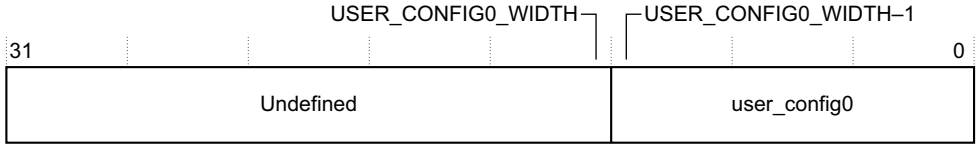


Figure 3-44 user_config0 Register bit assignments

Table 3-40 shows the user_config0 Register bit assignments.

Table 3-40 user_config0 Register bit assignments

Bits	Name	Function
[31:USER_CONFIG0_WIDTH[a]]	-	Undefined, write as zero.
[USER_CONFIG0_WIDTH–1:0]	user_config0	Sets the state of the **user_config0[USER_CONFIG0_WIDTH–1:0]** outputs.

a. If USER_CONFIG0_WIDTH is configured as 32 then none of the bits in this register are undefined.

3.3.38 User Config 1 Register

The user_config1 Register characteristics are:

Purpose Controls the state of the **user_config1** outputs.

Usage constraints No usage constraints.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-45 shows the user_config1 Register bit assignments.

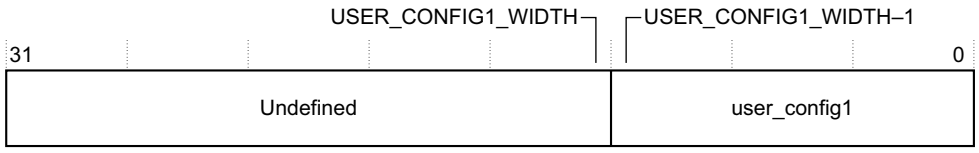


Figure 3-45 user_config1 Register bit assignments

Table 3-41 shows the user_config1 Register bit assignments.

Table 3-41 user_config1 Register bit assignments

Bits	Name	Function
[31:USER_CONFIG1_WIDTH[a]]	-	Undefined, write as zero.
[USER_CONFIG1_WIDTH–1:0]	user_config1	Sets the state of the **user_config1[USER_CONFIG1_WIDTH–1:0]** outputs.

a. If USER_CONFIG1_WIDTH is configured as 32 then none of the bits in this register are undefined.

3.3.39 Feature Control Register

The feature_ctrl Register characteristics are:

Purpose Controls the:
- early write response behavior
- write burst behavior.

Usage constraints Only accessible in Config or Low_power state.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-46 shows the feature_ctrl Register bit assignments.

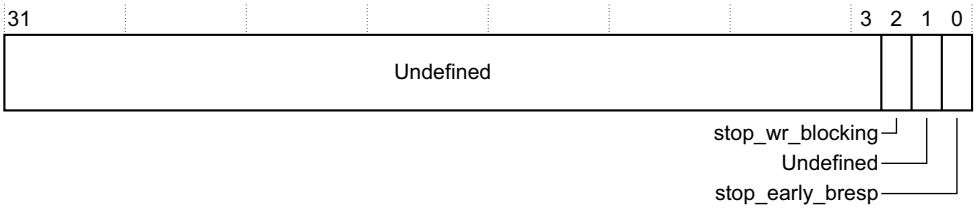


Figure 3-46 feature_ctrl Register bit assignments

Table 3-42 shows the feature_ctrl Register bit assignments.

Table 3-42 feature_ctrl registers bit assignments

Bits	Name	Function
[31:3]	-	Read undefined, write as zero.
[2]	stop_wr_blocking	Controls the write merge up to memory burst: 0 = write transfers initiated when a full write memory burst is ready 1 = write transfers initiated when write data is available.
[1]	-	Read undefined, write as zero.
[0]	stop_early_bresp	Controls the early write response feature: 0 = early write response enabled 1 = early write response disabled.

3.3.40 Integration Configuration Register

The int_cfg Register characteristics are:

Purpose Controls the enabling of the integration test logic.

Usage constraints Only accessible in Config state. ARM recommends that it is only accessed for integration testing or production testing.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-47 shows the int_cfg Register bit assignments.

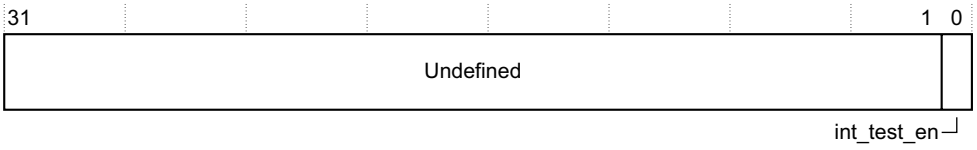


Figure 3-47 int_cfg Register bit assignments

Table 3-43 shows the int_cfg Register bit assignments.

Table 3-43 int_cfg Register bit assignments

Bits	Name	Function
[31:1]	-	Read undefined, write as zero.
[0]	int_test_en	Enables the integration test logic: 0 = disables the integration test logic 1 = enables the integration test logic. **Note** When the controller exits integration test mode, the **cactive** and **csysack** signals must be HIGH, even if the SoC does not use these signals. To satisfy this requirement you must program the int_outputs Register, see *Integration Outputs Register* on page 3-52.

3.3.41 Integration Inputs Register

The int_inputs Register characteristics are:

Purpose Provides the status of the following inputs:
- **csysreq**
- **qos_override[15:0]**.

Usage constraints
- Only accessible in Config state.
- Integration test logic must be enabled otherwise it ignores writes and reads return `0x0`. To enable the integration test logic, see *Integration Configuration Register* on page 3-50.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-48 shows the int_inputs Register bit assignments.

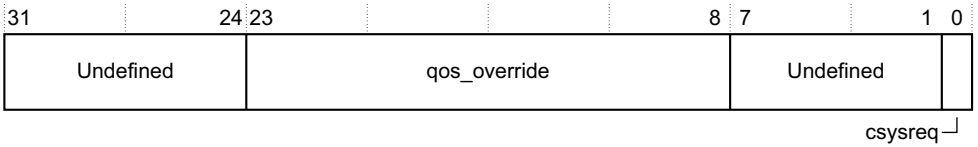


Figure 3-48 int_inputs Register bit assignments

Table 3-44 shows the int_inputs Register bit assignments.

Table 3-44 int_inputs Register bit assignments

Bits	Name	Function
[31:24]	-	Read undefined.
[23:8]	qos_override	Returns the status of the **qos_override[15:0]** inputs.
[7:1]	-	Read undefined.
[0]	csysreq	Returns the status of the **csysreq** input.

3.3.42 Integration Outputs Register

The int_outputs Register characteristics are:

Purpose Enables an external master to control the state of the following outputs:
- **cactive**
- **csysack**.

Usage constraints
- Only accessible in Config state.
- Integration test logic must be enabled otherwise it ignores writes and reads return `0x0`. To enable the integration test logic, see *Integration Configuration Register* on page 3-50.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

Figure 3-49 shows the int_outputs Register bit assignments.

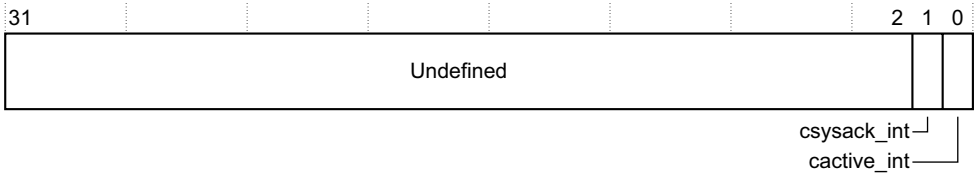


Figure 3-49 int_outputs Register bit assignments

Table 3-45 shows the int_outputs Register bit assignments.

Table 3-45 int_outputs Register bit assignments

Bits	Name	Function
[31:2]	-	Undefined, write as zero.
[1]	csysack_int	Controls the state of the **csysack** output.
[0]	cactive_int	Controls the state of the **cactive** output.

3.3.43 Peripheral Identification registers

The periph_id_[3:0] Register characteristics are:

Purpose Provide information about the configuration and version of the peripheral.

Usage constraints No usage constraints.

Configurations Available in all configurations of the LPDDR2 DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

These registers can conceptually be treated as a single register that holds a 32-bit peripheral ID value. Figure 3-50 shows the correspondence between bits [7:0] of the periph_id registers and the conceptual 32-bit Peripheral ID Register.

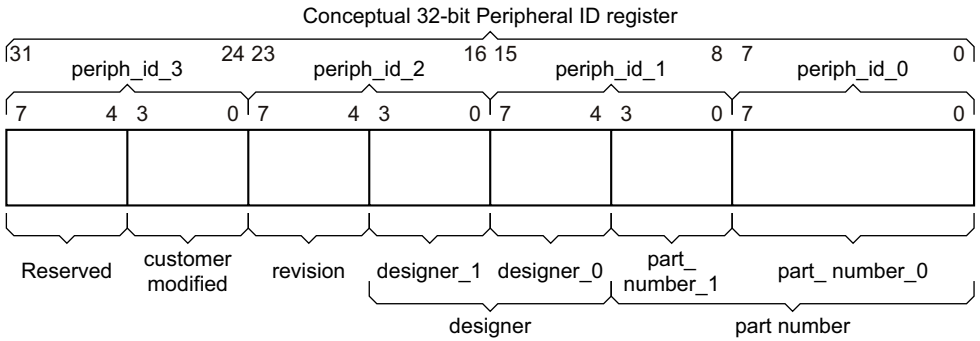


Figure 3-50 periph_id_[3:0] Register bit assignments

Table 3-46 shows the register bit assignments for the conceptual 32-bit peripheral ID register.

Table 3-46 Conceptual peripheral ID Register bit assignments

Bits	Name	Function
[31:28]	-	Reserved.
[27:24]	customer modified	Identifies data that is relevant to an ARM partner.
[23:20]	revision	Identifies the RTL revision of the peripheral.
[19:12]	designer	Identifies the designer. This is `0x41` for ARM.
[11:0]	part_number	Identifies the peripheral. The part number for the LPDDR2 DMC is `0x342`.

The following sections describe the periph_id registers:

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

Peripheral Identification Register 0

The periph_id_0 Register is hard-coded and the fields in the register control the reset value. Table 3-47 shows the register bit assignments.

Table 3-47 periph_id_0 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined.
[7:0]	part_number_0	Returns `0x42`.

Peripheral Identification Register 1

The periph_id_1 Register is hard-coded and the fields in the register control the reset value. Table 3-48 shows the register bit assignments.

Table 3-48 periph_id_1 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined.
[7:4]	designer_0	Returns `0x1`.
[3:0]	part_number_1	Returns `0x3`.

Peripheral Identification Register 2

The periph_id_2 Register is hard-coded and the fields in the register control the reset value. Table 3-49 shows the register bit assignments.

Table 3-49 periph_id_2 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined.
[7:4]	revision	These bits return the revision number: `0x0` = r0p0.
[3:0]	designer_1	Returns `0x4`.

Peripheral Identification Register 3

The periph_id_3 Register is hard-coded and the fields in the register control the reset value. Table 3-50 shows the register bit assignments.

Table 3-50 periph_id_3 Register bit assignments

Bits	Name	Function
[31:8]	-	Read undefined.
[7:4]	-	Reserved for future use, read undefined.
[3:0]	customer modified	Customer modified number, `0x0` from ARM.

3.3.44 Component Identification registers

The pcell_id_[3:0] Register characteristics are:

Purpose When concatenated, these four registers return `0xB105F00D` to

Usage constraints No usage constraints.

Configurations Available in all configurations of the DMC.

Attributes See the register summary in Table 3-1 on page 3-7.

These registers can be treated conceptually as a single register that holds a 32-bit component identification value. You can use the register for automatic BIOS configuration.

Figure 3-51 on page 3-56 shows the register bit assignments.

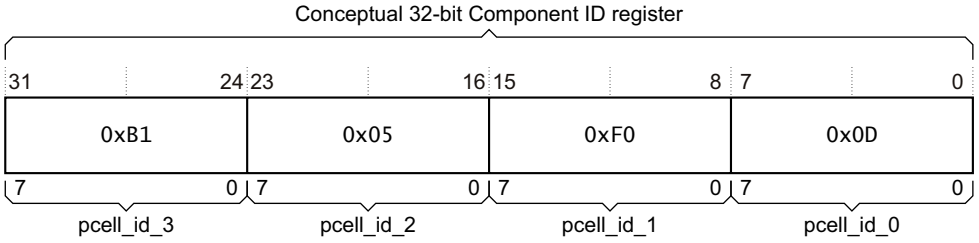


Figure 3-51 pcell_id Register bit assignments

Table 3-51 shows the register bit assignments.

Table 3-51 pcell_id Register bit assignments

pcell_id Register		**pcell_id_[3:0] registers**		
Bits	**Reset value**	**Register**	**Bits**	**Description**
[31:24]	0xB1	pcell_id_3	[31:8]	Read undefined.
			[7:0]	Returns 0xB1.
[23:16]	0x05	pcell_id_2	[31:8]	Read undefined.
			[7:0]	Returns 0x05.
[15:8]	0xF0	pcell_id_1	[31:8]	Read undefined.
			[7:0]	Returns 0xF0.
[7:0]	0x0D	pcell_id_0	[31:8]	Read undefined.
			[7:0]	Returns 0x0D.

Appendix A
Signal Descriptions

This appendix describes the LPDDR2 DMC signals. It contains the following sections:

A.1 Clock and reset signals

Table A-1 shows the clock and reset signals.

Table A-1 Clock and reset signals

Signal	Type	Source	Description
aclk	Input	Clock source	Clock for the **aclk** domain.
aresetn	Input	Reset source	**aclk** domain reset signal. This signal is active LOW.
cclken	Input	Bus clock	Clock enable for the AXI low-power interface.
mclk	Input	Clock source	Clock for **mclk** domain.
mresetn	Input	Reset source	Reset for **mclk** domain. This signal is active LOW.
pclken	Input	Bus clock	Clock enable for the APB interface.

A.2 Miscellaneous signals

This section describes the following miscellaneous signals:

- *QoS*
- *Tie-offs*
- *User signals* on page A-4
- *Scan test* on page A-5.

A.2.1 QoS

Table A-2 shows the QoS signal.

Table A-2 QoS signal

Signal	Type	Source	Description
qos_override[15:0]	Input	External control logic	When one or more bits are HIGH, coincident with **arvalid** and **arready**, and when the **arid** match bits are equivalent to the **qos_override** bit(s), then the QoS for the read access is forced to minimum latency. For more information, see *Quality of Service* on page 2-18.

A.2.2 Tie-offs

Table A-3 shows the tie-off signals.

Table A-3 Tie-off signals

Signal	Type	Source	Description
bank_bits[1:0]	Input	Tie-off	When **aresetn** is deasserted, the state of this signal sets the value of the bank_bits field in the memory_cfg2 Register. For more information about the values that a configured controller permits, see Memory Configuration 2 Register on page 3-29.
cke_init	Input	Tie-off	When **aresetn** is deasserted, the state of this signal sets the value of the cke_init bit in the memory_cfg2 Register. See *Memory Configuration 2 Register* on page 3-29.

Table A-3 Tie-off signals (continued)

Signal	Type	Source	Description
dfi_pipeline_bypass	Input	Tie-off	When set the LPDDR2 DMC bypasses the register stage in the DFI pad interface.
memory_width[1:0]	Input	Tie-off	When aresetn is deasserted, the state of this signal sets the value of the memory_width2 field in the memory_cfg2 Register. For information about the values that a configured controller permits, see Memory Configuration 2 Register on page 3-29.
sync	Input	Tie-off	When **aresetn** is deasserted, the state of this signal sets the value of the clock_cfg [0] bit in the memory_cfg2 Register. See *Memory Configuration 2 Register* on page 3-29. Set this signal HIGH if **aclk** is synchronous to **mclk**. Set this signal LOW if **aclk** is asynchronous to **mclk**.

A.2.3 User signals

User signals are general purpose I/O signals that you can use to control external devices, such as a *Delay-Locked Loop* (DLL). Table A-4 shows the user signals.

Table A-4 User signals

Signal	Type	Source or destination	Description
user_config0[USER_CONFIG0_WIDTH–1:0]	Output	External control logic	General purpose control signals that you program using the *User Config 0 Register* on page 3-47
user_config1[USER_CONFIG1_WIDTH–1:0]	Output	External control logic	General purpose control signals that you program using the *User Config 1 Register* on page 3-48
user_status[USER_STATUS_WIDTH–1:0]	Input	External control logic	General-purpose input signals that are read using the *User Status Register* on page 3-46

A.2.4 Scan test

Table A-5 shows the scan test signals.

Table A-5 Scan test signals

Signal	Type	Source	Description
rst_bypass	Input	Tie-off	This signal is used for ATPG testing only
se	Input	Tie-off	This signal enables scan mode

A.3 AXI signals

The following sections list the AXI signals:

- *Write address channel signals*
- *Write data channel signals* on page A-7
- *Write response channel signals* on page A-7
- *Read address channel signals* on page A-8
- *Read data channel signals* on page A-9
- *AXI low-power interface signals* on page A-9.

A.3.1 Write address channel signals

Table A-6 shows the AXI write address channel signals.

Table A-6 Write address channel signals

Signal	AMBA equivalent[a]
awaddr[31:0]	**AWADDR**
awburst[1:0]	**AWBURST[1:0]**
awcache[3:0][b]	**AWCACHE[3:0]**
awid[AID_WIDTH–1:0][c]	**AWID**
awlen[3:0]	**AWLEN[3:0]**
awlock[1:0]	**AWLOCK[1:0]**
awprot[2:0][b]	**AWPROT[2:0]**
awready	**AWREADY**
awsize[2:0]	**AWSIZE[2:0]**
awvalid	**AWVALID**

a. For a description of these signals, see the *AMBA AXI v1.0 Protocol Specification*.
b. The LPDDR2 DMC ignores any information that it receives on these signals.
c. The value of **AID_WIDTH** is set during configuration of the LPDDR2 DMC.

A.3.2 Write data channel signals

Table A-7 shows the AXI write data channel signals.

Table A-7 Write data channel signals

Signal	AMBA equivalent[a]
wdata[AXI_DATA_MSB:0][b]	**WDATA**
wid[AID_WIDTH–1:0][b]	**WID**
wlast	**WLAST**
wready	**WREADY**
wstrb[AXI_STRB_MSB:0][b]	**WSTRB**
wvalid	**WVALID**

a. For a description of these signals, see the *AMBA AXI v1.0 Protocol Specification*.
b. The value of **AXI_DATA_MSB** and **AID_WIDTH** are set during configuration of the LPDDR2 DMC. **AXI_STRB_MSB**=**AXI_DATA_MSB**÷8.

A.3.3 Write response channel signals

Table A-8 shows the AXI write response channel signals.

Table A-8 Write response channel signals

Signal	AMBA equivalent[a]
bid[AID_WIDTH–1:0][b]	**BID**
bready	**BREADY**
bresp[1:0][c]	**BRESP[1:0]**
bvalid	**BVALID**

a. For a description of these signals, see the *AMBA AXI v1.0 Protocol Specification*.
b. The value of **AID_WIDTH** is set during configuration of the LPDDR2 DMC.
c. The LPDDR2 DMC ties **bresp[1]** LOW and therefore it only provides OKAY or EXOKAY responses.

A.3.4 Read address channel signals

Table A-9 shows the AXI read address channel signals.

Table A-9 Read address channel signals

Signal	AMBA equivalent[a]
araddr[31:0]	**ARADDR**
arburst[1:0]	**ARBURST[1:0]**
arcache[3:0][b]	**ARCACHE[3:0]**
arid[AID_WIDTH–1:0][c]	**ARID**
arlen[3:0]	**ARLEN[3:0]**
arlock[1:0]	**ARLOCK[1:0]**
arprot[2:0][b]	**ARPROT[2:0]**
arready	**ARREADY**
arsize[2:0]	**ARSIZE[2:0]**
arvalid	**ARVALID**

a. For a description of these signals, see the *AMBA AXI v1.0 Protocol Specification*.
b. The LPDDR2 DMC ignores any information that it receives on these signals.
c. The value of **AID_WIDTH** is set during configuration of the LPDDR2 DMC.

A.3.5 Read data channel signals

Table A-10 shows the AXI read data channel signals.

Table A-10 Read data channel signals

Signal	AMBA equivalent[a]
rdata[AXI_DATA_MSB:0][b]	**RDATA**
rid[AID_WIDTH–1:0][b]	**RID**
rlast	**RLAST**
rready[c]	**RREADY**
rresp[1:0][d]	**RRESP[1:0]**
rvalid	**RVALID**

a. For a description of these signals, see the *AMBA AXI v1.0 Protocol Specification*.
b. The value of **AXI_DATA_MSB** and **AID_WIDTH** are set during configuration of the LPDDR2 DMC.
c. It is possible for refreshes to be missed if **rready** is held LOW for longer than one refresh period, and the read data FIFO, command FIFO, and arbiter queue become full. An OVL error is triggered if this occurs in simulation. Ensure that the controller has a sufficiently high system priority to prevent this.
d. The LPDDR2 DMC ties **rresp[1]** LOW and therefore it only provides OKAY or EXOKAY responses.

A.3.6 AXI low-power interface signals

Table A-11 shows the AXI low-power interface signals.

Table A-11 AXI low-power interface signals

Signal	AMBA equivalent[a]
cactive	**CACTIVE**
csysack	**CSYSACK**
csysreq	**CSYSREQ**

a. For a description of these signals, see the *AMBA AXI v1.0 Protocol Specification*.

A.4 APB signals

Table A-12 shows the APB signals.

Table A-12 APB interface signals

Signal	AMBA equivalent[a]
paddr[31:0][b]	**PADDR**
penable	**PENABLE**
prdata[31:0]	**PRDATA**
pready	**PREADY**
psel	**PSELx**
pslverr[c]	**PSLVERR**
pwdata[31:0]	**PWDATA**
pwrite	**PWRITE**

a. For a description of these signals, see the *AMBA 3 APB Protocol Specification.*
b. The LPDDR2 DMC uses bits [11:2]. It ignores bits [31:12] and bits [1:0].
c. The LPDDR2 DMC ties **pslverr** LOW.

A.5 DFI pad interface signals

Table A-13 shows the DFI pad interface signals.For a description of these signals, see the *DDR PHY Interface (DFI) Specification*.

Table A-13 DFI pad interface signals

Signal	Type	Source or destination
dfi_address[15:0]	Output	PHY device
dfi_bank[MEMORY_BANK_WIDTH–1:0][a]	Output	
dfi_cas_n	Output	
dfi_cke[MEMORY_CHIPS–1:0][b]	Output	
dfi_cs_n[MEMORY_CHIPS–1:0][b]	Output	
dfi_dram_clk_disable[MEMORY_CHIPS–1:0][b]	Output	
dfi_init_complete	Input	
dfi_init_start	Output	
dfi_phyupd_ack	Output	
dfi_phyupd_req	Input	
dfi_phyupd_type[1:0]	Input	
dfi_ras_n	Output	
dfi_we_n	Output	
dfi_rddata[2×MEMWIDTH–1:0][c]	Input	
dfi_rddata_en[2×MEMORY_BYTES–1:0][d]	Output	
dfi_wrdata_en[2×MEMORY_BYTES–1:0][d]	Output	
dfi_wrdata[2×MEMWIDTH–1:0][c]	Output	
dfi_wrdata_mask[2×MEMORY_BYTES–1:0][d]	Output	

a. The MEMORY_BANK_WIDTH value depends on the number of banks in the configuration.
b. MEMORY_CHIPS is the number of chip selects and is set during configuration of the LPDDR2 DMC.
c. MEMWIDTH is the width of the external memory data bus in bits and is set during configuration of the LPDDR2 DMC.

d. MEMORY_BYTES is the width of the external memory data bus in bytes and is set during configuration of the LPDDR2 DMC.

Appendix B
Revisions

This appendix describes the technical changes between released issues of this book.

Table B-1 Issue A

Change	Location	Affects
No changes, first release	-	-

Glossary

This glossary describes some of the terms used in technical documents from ARM.

Advanced eXtensible Interface (AXI)
A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure.

The AXI protocol also includes optional extensions to cover signaling for low-power operation.

AXI is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

Advanced Microcontroller Bus Architecture (AMBA)
A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that describes a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

Advanced Peripheral Bus (APB)
A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

AMBA *See* Advanced Microcontroller Bus Architecture.

APB *See* Advanced Peripheral Bus.

Architecture The organization of hardware and/or software that characterizes a processor and its attached components, and enables devices with similar characteristics to be grouped together when describing their behavior, for example, Harvard architecture, instruction set architecture, ARMv6 architecture.

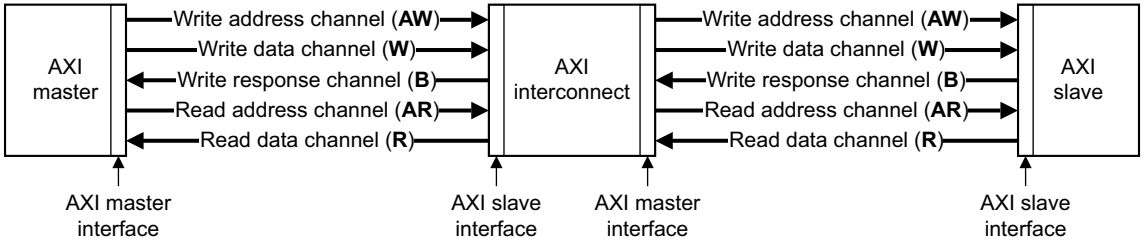
ATPG *See* Automatic Test Pattern Generation.

Automatic Test Pattern Generation (ATPG)
The process of automatically generating manufacturing test vectors for an ASIC design, using a specialized software tool.

AXI *See* Advanced eXtensible Interface.

AXI channel order and interfaces
The block diagram shows:

- the order in which AXI channel signals are described
- the master and slave interface conventions for AXI components.



AXI terminology The following AXI terms are general. They apply to both masters and slaves:

Active read transaction

A transaction for which the read address has transferred, but the last read data has not yet transferred.

Active transfer

A transfer for which the **xVALID**[1] handshake has asserted, but for which **xREADY** has not yet asserted.

Active write transaction

A transaction for which the write address or leading write data has transferred, but the write response has not yet transferred.

Completed transfer

A transfer for which the **xVALID**/**xREADY** handshake is complete.

Payload The non-handshake signals in a transfer.

Transaction An entire burst of transfers, comprising an address, one or more data transfers and a response transfer (writes only).

Transmit An initiator driving the payload and asserting the relevant **xVALID** signal.

Transfer A single exchange of information. That is, with one **xVALID**/**xREADY** handshake.

The following AXI terms are master interface attributes. To obtain optimum performance, they must be specified for all components with an AXI master interface:

Combined issuing capability

The maximum number of active transactions that a master interface can generate. It is specified for master interfaces that use combined storage for active write and read transactions. If not specified then it is assumed to be equal to the sum of the write and read issuing capabilities.

Read ID capability

The maximum number of different **ARID** values that a master interface can generate for all active read transactions at any one time.

Read ID width

The number of bits in the **ARID** bus.

Read issuing capability

The maximum number of active read transactions that a master interface can generate.

---

1. The letter **x** in the signal name denotes an AXI channel as follows:

AW Write address channel.
W Write data channel.
B Write response channel.
AR Read address channel.
R Read data channel.

Write ID capability

The maximum number of different **AWID** values that a master interface can generate for all active write transactions at any one time.

Write ID width

The number of bits in the **AWID** and **WID** buses.

Write interleave capability

The number of active write transactions for which the master interface is capable of transmitting data. This is counted from the earliest transaction.

Write issuing capability

The maximum number of active write transactions that a master interface can generate.

The following AXI terms are slave interface attributes. To obtain optimum performance, they must be specified for all components with an AXI slave interface:

Combined acceptance capability

The maximum number of active transactions that a slave interface can accept. It is specified for slave interfaces that use combined storage for active write and read transactions. If not specified then it is assumed to be equal to the sum of the write and read acceptance capabilities.

Read acceptance capability

The maximum number of active read transactions that a slave interface can accept.

Read data reordering depth

The number of active read transactions for which a slave interface can transmit data. This is counted from the earliest transaction.

Write acceptance capability

The maximum number of active write transactions that a slave interface can accept.

Write interleave depth

The number of active write transactions for which the slave interface can receive data. This is counted from the earliest transaction.

Beat

Alternative word for an individual transfer within a burst. For example, an INCR4 burst comprises four beats.

See also Burst.

Burst
A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented.

See also Beat.

Clock gating
Gating a clock signal for a macrocell with a control signal, and using the modified clock that results to control the operating state of the macrocell.

Cold reset
Also known as power-on reset. Starting the processor by turning power on. Turning power off and then back on again clears main memory and many internal settings. Some program failures can lock up the processor and require a cold reset to enable the system to be used again. In other cases, only a warm reset is required.

See also Warm reset.

Halfword
A 16-bit data item.

Macrocell
A complex logic block with a defined interface and behavior. A typical VLSI system comprises several macrocells (such as a processor, an ETM, and a memory block) plus application-specific logic.

Memory bank
One of two or more parallel divisions of interleaved memory, usually one word wide, that enable reads and writes of multiple words at a time, rather than single words. All memory banks are addressed simultaneously and a bank enable or chip select signal determines which of the banks is accessed for each transfer. Accesses to sequential word addresses cause accesses to sequential banks. This enables the delays associated with accessing a bank to occur during the access to its adjacent bank, speeding up memory transfers.

Power-on reset
See Cold reset.

Processor
A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

Region
A partition of instruction or data memory space.

Reserved
A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

Scan chain A scan chain is made up of serially-connected devices that implement boundary scan technology using a standard JTAG TAP interface. Each device contains at least one TAP controller containing shift registers that form the chain connected between **TDI** and **TDO**, through which test data is shifted. Processors can contain several shift registers to enable you to access selected parts of the device.

Unpredictable For reads, the data returned when reading from this location is unpredictable. It can have any value. For writes, writing to this location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor, or any part of the system.

Warm reset Also known as a core reset. Initializes the majority of the processor excluding the debug controller and debug logic. This type of reset is useful if you are using the debugging features of a processor.

Word A 32-bit data item.