PrimeCell® Dynamic Memory Controller (PL340)

Revision: r2p0

Technical Reference Manual

ARM®

ARM DDI 0331E

PrimeCell Dynamic Memory Controller (PL340)
Technical Reference Manual

Release Information

Change history

Date	Issue	Confidentiality	Change
22 June 2004	A	Non-Confidential	First release for r0p0.
31 August 2004	B	Non-Confidential	Second release for r0p0.
25 August 2005	C	Non-Confidential	Incorporate erratum. Additional information to *Exclusive access* on page 2-6.
09 June 2006	D	Non-Confidential	First release for r1p0.
16 May 2007	E	Non-Confidential	First release for r2p0.

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.

Product Status

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

Web Address

`http://www.arm.com`

Contents
PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual

List of Tables
PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual

List of Figures
PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual

Preface

This preface introduces the *PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual*. It contains the following sections:

About this manual

This is the Technical Reference Manual for the PrimeCell *Dynamic Memory Controller* (DMC). The memory controller is highly configurable to support multiple types and sizes of SDRAM.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this manual, 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 manual is written for implementation engineers and architects, and provides a description of an optimal memory controller architecture. The memory controller provides an interface between the *Advanced eXtensible Interface* (AXI) system bus and external, off-chip, memory devices.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the memory controller and its features.

Chapter 2 *Functional Overview*

Read this chapter for an overview of the major functional blocks and the operation of the memory controller.

Chapter 3 *Programmer's Model*

Read this chapter for a description of the registers.

Chapter 4 *Programmer's Model for Test*

Read this chapter for a description of the additional logic for integration testing.

Chapter 5 *Device Driver*

Read this chapter for a description of the device driver functions.

Appendix A *Signal Descriptions*

Read this appendix for a description of the memory controller signals.

Glossary

Read the Glossary for definitions of terms used in this manual.

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams* on page xii
- *Signals* on page xii
- *Numbering* on page xiii.

Typographical

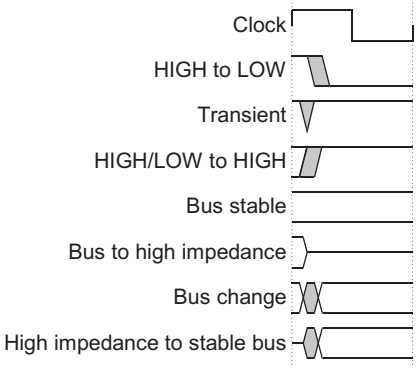
The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`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 >	Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example: • `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>` • The Opcode_2 value selects which register is accessed.

Timing diagrams

The figure named *Key to timing diagram conventions* 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 and LOW for active-LOW signals.

Lower-case n Denotes an active-LOW signal.

Prefix A Denotes global *Advanced eXtensible Interface* (AXI) signals:

Prefix AR Denotes AXI read address channel signals.

Prefix AW Denotes AXI write address channel signals.

Prefix B Denotes AXI write response channel signals.

Prefix C Denotes AXI low-power interface signals.

Prefix H Denotes *Advanced High-performance Bus* (AHB) signals.

Prefix P Denotes Advanced Peripheral Bus (APB) signals.

Prefix R Denotes AXI read data channel signals.

Prefix W Denotes AXI write data channel signals.

Numbering

The numbering convention is:

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

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

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

Further reading

This section lists publications by ARM Limited.

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

ARM publications

This manual contains information that is specific to the memory controller. See the following documents for other relevant information:

- *PrimeCell Dynamic Memory Controller (PL340) Integration Manual* (ARM DII 0105)
- *PrimeCell Dynamic Memory Controller (PL340) Implementation Guide* (ARM DII 0106)
- *AMBA Designer (FD001) User Guide* (ARM DUI 0333)
- *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual* (ARM DDI 0249)
- *AMBA AXI Protocol Specification* (ARM IHI 0022)

- *AMBA 3 APB Protocol Specification* (ARM IHI 0024).

Feedback

ARM Limited welcomes feedback on the memory controller and its documentation.

Feedback on this memory controller

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

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

Feedback on this manual

If you have any comments on this manual, send e-mail to `errata@arm.com` giving:

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

ARM Limited also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the memory controller and contains the following sections:

1.1 About the memory controller

The memory controller is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant PrimeCell peripheral that is developed, tested, and licensed by ARM.

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

You can configure the memory controller with a number of options, for example:

- the SDRAM memory type, see Table 1-1 on page 1-4
- the number of SDRAM memory devices
- the maximum SDRAM memory width
- the number of outstanding AXI addresses.

For a complete list of the configurable options, see *Features of the memory controller* on page 1-3.

Note

See *Product revisions* on page 1-8, for differences between revisions.

The memory controller supports the PrimeCell (PL220) *External Bus Interface* (EBI). This ensures that you can still use a shared external bus.

For more information on AMBA see:

- *AMBA AXI Protocol Specification*
- *AMBA 3 APB Protocol Specification*.

Figure 1-1 on page 1-3 shows an example memory controller system.

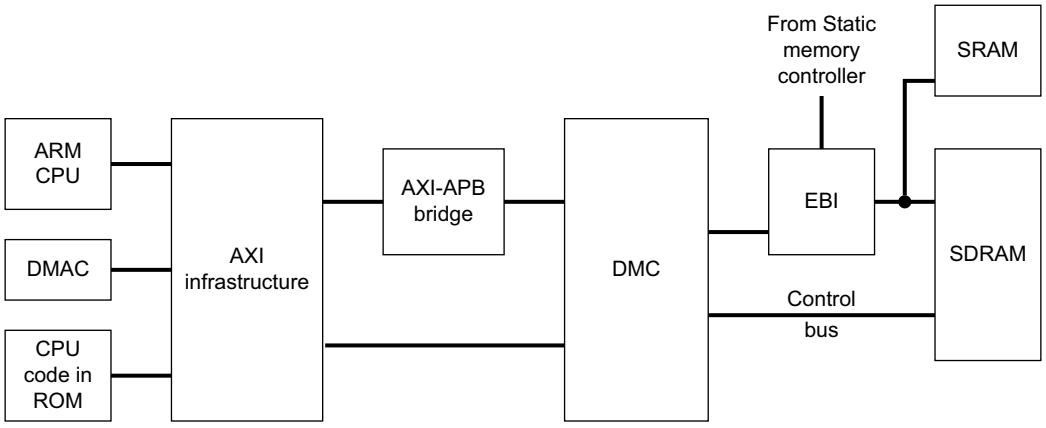


Figure 1-1 Example system

1.1.1 Features of the memory controller

The memory controller block offers the following features:

- configurable, soft macrocell available in Verilog
- scalable pipeline
- interface between AMBA AXI bus fabric and DDR, LPDDR, SDR and eDRAM memories
- Quality of Service (QoS) and request arbitration features for low latency transfers and optimal use of memory bandwidth
- packing and unpacking of memory data access for AXI transactions width less than the memory width
- write data interleaving supported
- multiple outstanding addresses supported
- support for ARMv6 outstanding exclusive accesses
- support synchronous and asynchronous operation between AXI bus fabric and external memory bus
- programmable support for memory power saving modes including *Deep Power Down* (DPD), active power-down, precharge power-down and self-refresh
- programmable through AMBA APB interface

- synchronous n:1 between AXI and APB
- area/performance optimization, trade-offs through configurable hardware resources
- optimized utilization of external memory bus
- one to four external chip selects
- configurable for single port for all chip selects on individual **cke** port per external chip select.

1.1.2 Supported memory widths

Table 1-1 lists the supported memory bus widths for different memory types.

Table 1-1 Memory widths

Memory bus width	SDR	Mobile DDR/DDR
16-bit	Yes	Yes
32-bit	Yes	Yes
64-bit	Yes	Yes

Table 1-2 lists the supported memory widths for different AXI interface widths.

Table 1-2 Memory widths and AXI port widths

Effective memory width[a]	AXI port width		
	32-bit	64-bit	128-bit
16-bit	Yes	No	No
32-bit	Yes	Yes	No
64-bit	Yes	Yes	Yes
128-bit	No	Yes	Yes

a. Effective memory width is the data transferred in a single clock cycle, that is, 32 bits for 32-bit SDR and 32bits for 16-bit DDR.

Note

All memory configurations support 1-4 external memory chip selects.

You can use each memory controller configuration at half of its memory width, by setting the memory width tie-off or programming through the memory_cfg2 Register, provided that:

- the new memory width is not less than 16 bits
- the effective memory width is not less than half the AXI interface width.

1.1.3 AXI interface attributes

The slave interface has the following attributes:

Write Acceptance Capability

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

Read Acceptance Capability

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

Combined Acceptance Capability

The maximum number of active transactions that a slave can accept. This is indicated where the slave has combined read and write transaction storage so this is mutually exclusive to the write or read acceptance capability.

Write Interleave Depth

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

Read Data Reordering Depth

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

Table 1-3 lists the attribute formats.

Table 1-3 Attribute formats

Attribute	Value
Combined acceptance capability	Arbiter queue depth
Write interleave depth	Write acceptance capability
Read data reorder depth	Read acceptance capability

1.2 Supported devices

See the product *Release Note* for more information.

1.3 Product revisions

This is the Technical Reference Manual, for the PrimeCell Dynamic Memory Controller (PL340), r2p0. See *Product revision status* on page x for a description of revision numbering. This section summarizes the differences in functionality between the releases of these products.

r0p0 - r1p0 This release includes

- A new configurable part.

r1p0 - r2p0 This release includes:

- New configurable options:
  - configuration that is programmable to support SDR, (LP)DDR memory types
  - eDRAM configuration
  - a global **cke** port or a local **cke** port for each memory device
  - scalable pipeline
  - low-power modes of operation.
- New programmable options:
  - DPD support
  - variable number of auto-refresh requests before priority change.
- New functionality
  - improved write termination
  - programmable read_delay.

Chapter 2
Functional Overview

This chapter describes the memory controller operation. It contains the following sections:

2.1 Memory controller overview

Figure 2-1 shows the interfaces of the memory controller.

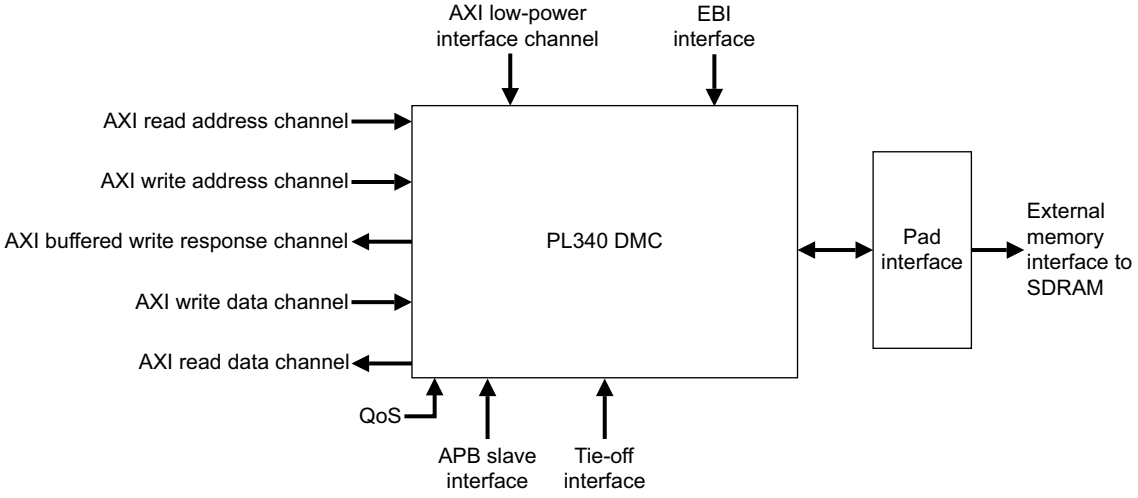


Figure 2-1 DMC block diagram

2.2 AXI slave interface

For more information about the AXI interface, see the *AMBA AXI Protocol Specification*. The AXI slave interface comprises the following AXI channels:

AXI write address channel

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

AXI write data channel

Enables the transfer of write data and validating data byte strobes to the memory controller.

AXI buffered write response channel

Enables the transfer of response information associated with a write transaction and a write data channel transaction.

AXI read address channel

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

AXI read data channel

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

Figure 2-2 on page 2-4 shows the AXI slave interface external connections.

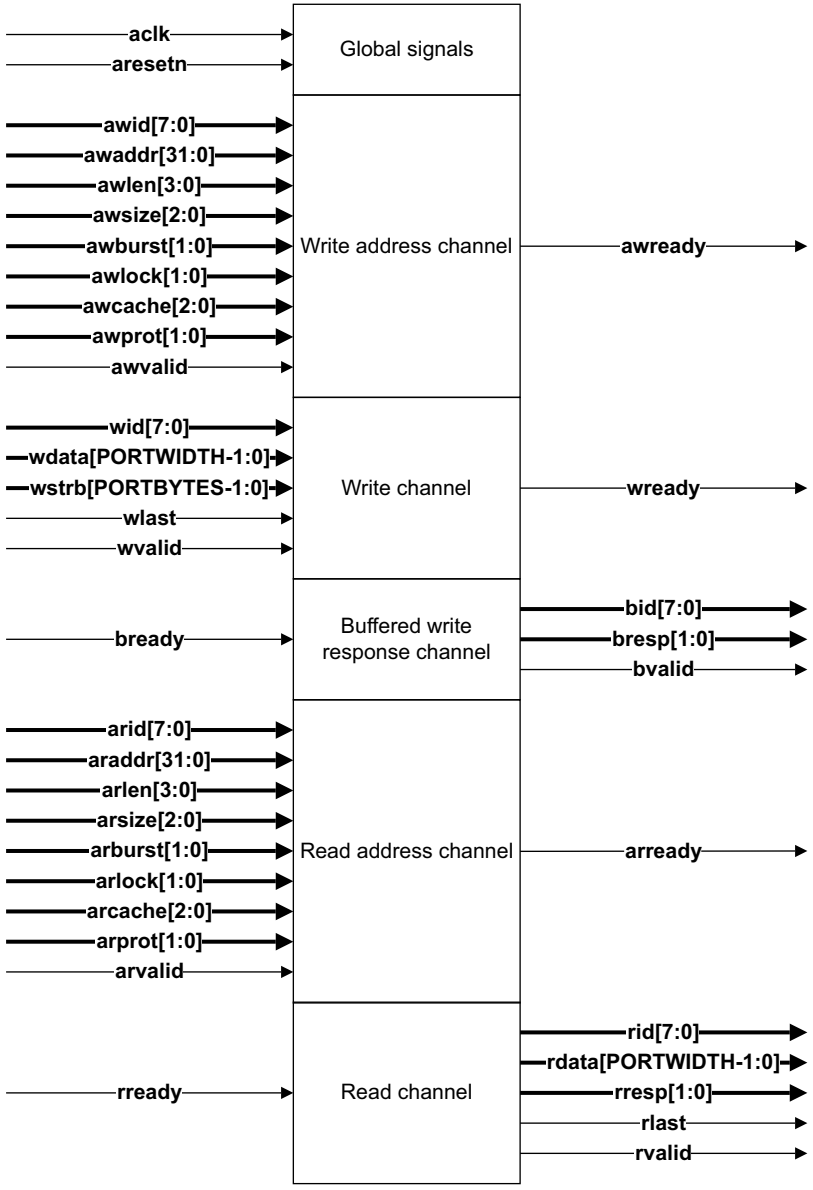


Figure 2-2 AXI slave interface connections

Note

Refreshes can be missed if **rready** is held LOW, or the *External Bus Interface* (EBI) is not granted, for longer than one refresh period and the read data FIFO, command FIFO, and the arbiter queue becomes full. An OVL error triggers if this occurs in simulation. Ensure that the device has a sufficiently high system priority to prevent this.

The AXI programmer's view is of a flat area of memory. The full range of AXI operations are supported provided that the memory burst length selected is less than the read data FIFO depth.

Note

- The FIFO depths are set as part of the configuration process. See the *AMBA Designer (FD001) User Guide* for more information.
- The **arcache** and **arprot** signals are included in the memory controller port list for completeness only. There is no requirement for the memory controller to use these signals.

AXI write transactions that have a write data width of less than the effective memory width, see Table 1-2 on page 1-4, have the data packed to ensure optimum memory accesses when possible.

AXI read transactions that have a read data width of less than the effective memory width, see Table 1-2 on page 1-4, have the data unpacked from optimum memory accesses.

Note

Figure 2-2 on page 2-4, Figure 2-3 on page 2-9, Figure 2-4 on page 2-10 and Figure 2-18 on page 2-20 do not show reset signals. See Table A-1 on page A-2 for more information.

2.2.1 Memory device base address

The base addresses of the external memory devices are programmable using the chip_cfg Registers. See *chip_<n>_cfg Registers* on page 3-26.

2.2.2 Formatting AXI address channel payload

Formatting is as follows:

Chip select decoding

Using the programmed values in the chip_<n>_cfg Registers that Chapter 3 *Programmer's Model* defines, an incoming address has the most significant eight address bits compared with the address match bits using the address mask to ignore any don't care bits to select an external chip.

The transfer is still carried out if there is no match, but the result is undefined.

Row select decoding

The AXI address determines the row address using bits [5:3] of the memory_cfg Register, and also the brc_n_rbc bit for the selected chip defined in the chip_<n>_cfg Register.

Column select decoding

The AXI address determines the column address using bits [2:0] of the memory_cfg Register.

Bank select decoding

The AXI address determine the chip bank depending on the configuration of the PL340 in addition to the brc_n_rbc bit for the selected chip defined in the chip_<n>_cfg register.

2.2.3 Exclusive access

In addition to reads and writes, exclusive reads and writes are supported in accordance with 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 data mask for the write is forced 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.

For the purposes of monitoring, address comparison is made using a bit mask derived in the following fashion.

Consider the byte addresses accessed by a transaction. All the least significant bits, up to and including, the most significant bit that vary between those addresses are set to logic zero in the mask. All the stable address bits above this point are set to logic one.

Example 2-1 provides information about three transactions.

Example 2-1

Exclusive Read	Address = `0x100`, size = WORD, length = 1, ID = 0.
Write	Address = `0x104`, size = WORD, length = 2, ID = 1.
Exclusive Write	Address = `0x100`, size = WORD, length = 1, ID = 0.

The write transaction accesses the address range `0x104-0x10B`. Therefore, address bit 3 is the most significant bit that varies between byte addresses. The bit mask is therefore formed so that address bits 3 down to 0 are not compared. This has the effect that the masked write, as far as the monitoring logic has calculated, has accessed the monitored address. Therefore the exclusive write is marked as having failed.

Table 2-1 lists the address comparison steps:

Table 2-1 Address comparison steps example

Step		Binary	Hex
1	Monitored address	b000100000000	`0x100`
2	Write address	b000100000100	`0x104`
3	Write accesses	b000100000100	`0x104`
		b000100000101	`0x105`
		b000100000110	`0x106`
		b000100000111	`0x107`
		b000100001000	`0x108`

Table 2-1 Address comparison steps example (continued)

Step		Binary	Hex
		b000100001001	0x109
		b000100001010	0x10A
		b000100001011	0x10B
4	Generate a comparison mask	b111111110000	0xFF0
5	Monitored address ANDed with mask	b000100000000	0x100
6	Write Address ANDed with mask	b000100000000	0x100
7	Compare steps 5 and 6		
8	Mark exclusive write as failed		

This example shows how the logic can introduce false-negatives in exclusive access monitoring, because in reality the write has not accessed the monitored address. The implementation has been chosen to reduce design complexity, but always provides safe behavior.

When calculating the address region accessed by the write, the burst type is always taken to be INCR. Therefore, a wrapped transaction in Example 2-1 on page 2-7 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.

2.3 AXI low-power interface

Figure 2-3 shows AXI low-power interface channel signals.

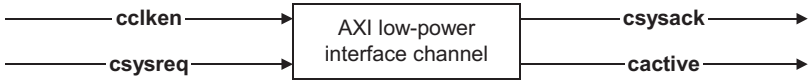


Figure 2-3 AXI low-power interface channel signals

For more information about the AXI Low-Power Interface, see the AMBA AXI Protocol Specification.

The low-power interface can move the memory controller into its Low-power state without the requirement for any register accesses, see also *aclk domain state diagram* on page 2-24 and *Low-power operation* on page 2-33.

If you do not require the low-power interface, tie it off as the *PrimeCell Dynamic Memory Controller (PL340) Integration Manual* describes.

2.4 APB slave interface

The APB interface is a fully compliant APB slave. The memory controller has 4KB of memory allocated to it.

See the *AMBA 3 APB Protocol Specification* for more information about the APB interface.

The APB interface implements the register file as Chapter 3 *Programmer's Model* describes.

Note

The APB only supports single-word 32-bit accesses. Bits [1:0] of the **paddr** signal are not used within the memory controller, resulting in byte and half-word accesses being treated as word accesses.

Figure 2-4 shows the APB external connections.

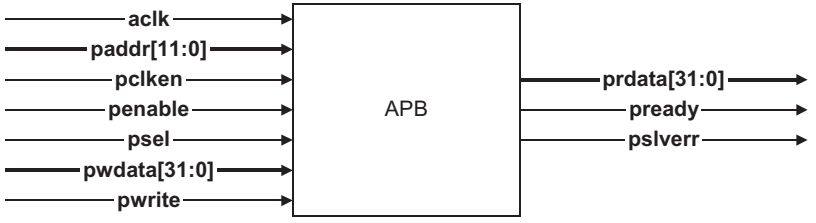


Figure 2-4 APB external connections

The APB interface enables the memory controller state of operation in addition to programming the memory controller with the correct timings and settings for the connected memory type. The APB interface also initializes the connected memory devices see *Initialization* on page 2-27.

The APB interface is clocked by the same clock as the AXI domain clock, **aclk**. However, the interface has also a clock enable, enabling it to be slowed down to execute at an integer divisor of **aclk**.

Note

The **pslverr** output is included for completeness, and the DMC permanently drives it LOW.

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
- when a memory command is received, and a previous memory command has not been completed.

Note

The **pslverr** output is included for completeness, it is permanently driven LOW by the DMC.

The only registers that can be accessed when the memory controller is not in the Config or Low-power state are the Memory Controller Status Register, to read the current state, and the Memory Controller Command Register, to change state.

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 memory controller is in the Low-power state. This is because the refresh rate depends on the **mclk** period. It is recommended that direct commands to the external memories are only written when the memory controller is in the Config state and not in the Low-power state.

2.5 Tie-off

You can use a number of signal ports to alter some configuration parameters of the controller. You can also change these parameters through the APB interface using the memory_cfg2 Register detailed in the programmers model.

The value of each tie-off port determines its respective bits in the register after a reset.

2.6 Memory interface

The memory interface provides a clean and defined interface between the 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 *Programmer's Model*.

The external I/Os to this block are:

mclk Clock for **mclk** domain.

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

ebibackoff Tie this LOW to indicate that the memory controller must not back off from the bus if you are not using an EBI.

ebigrant Tie this HIGH to indicate that the bus is always granted if you are not using an EBI.

ebireq Leave this unconnected if you are not using an EBI.

use_ebi Tie this LOW if you are not using an EBI.

The memory interface tracks and controls the state of the external memories using either an FSM per extended memory or one FSM depending on the configuration of the memory controller. Figure 2-5 shows the **mclk** domain FSM.

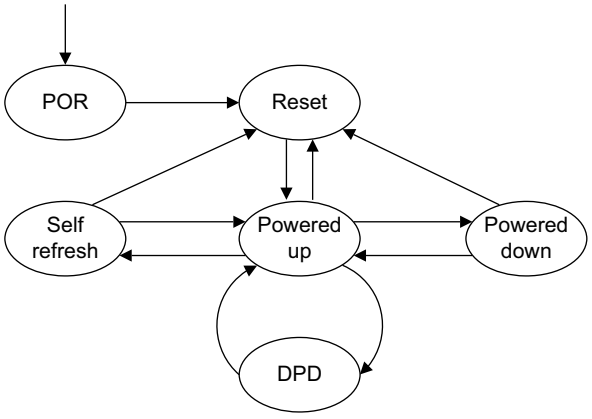


Figure 2-5 mclk domain FSM

See Table 2-3 on page 2-34 for valid system states and *Deep power-down* on page 2-42.

The interface is separated from the arbiter using the following configurable synchronous or asynchronous FIFOs:

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

There is also a static interface which has configuration signals that cannot be changed when the interface is operating.

The memory interface reads commands from the arbiter using a 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 FIFO cannot be executed, then the command pipeline stalls until it can be executed.

The timing parameters in Figure 2-6 on page 2-15 to Figure 2-17 on page 2-19 can all be programmed using the APB interface. See Chapter 3 *Programmer's Model*.

2.6.1 Memory interface to pad interface timing

All command control outputs are clocked on the same edge. In Figure 2-6 on page 2-15 to Figure 2-17 on page 2-19, the control outputs to the external memory are always clocked on the falling edge of the memory clock.

The relative times between control signals from the memory interface are maintained when output from the pad interface to the actual SDRAM devices. Therefore, the timing register values required for a particular SDRAM device can be determined from that SDRAM device's data sheet. Figure 2-6 on page 2-15 to Figure 2-17 on page 2-19 show how the data sheet timings map onto the memory controller timing registers.

The times in Figure 2-6 on page 2-15 to Figure 2-17 on page 2-19 are not necessarily the default timing values, but are values that are small enough to show the entire delay in one figure.

Note

- The **command_en**, **data_cntl_en**, and **read_en** signals are internal to the memory controller.
- The t_rcd, t_rfc and t_rp have an additional timing parameter called `schedule_rx`, defined in their registers. The schedule parameters prevent commands being issued that can stall the command pipeline and are timed in **aclk** cycles. For synchronized 1:1 operation of **aclk** and **mclk**, `schedule_rx` must be programmed

to t_rx -3. For non-synchronized 1:1 operation, the value must be scaled accordingly.

Figure 2-6 on page 2-15 shows the command control output timing.

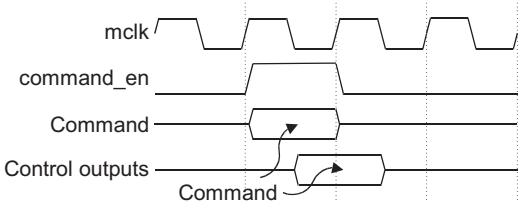


Figure 2-6 Command control output timing

Figure 2-7 shows the activate to read or write command timing.

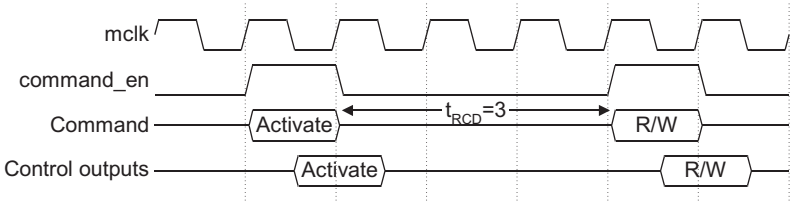


Figure 2-7 Activate to read or write command timing, t_{RCD}

Figure 2-8 shows the bank activate to bank activate or auto-refresh command timing.

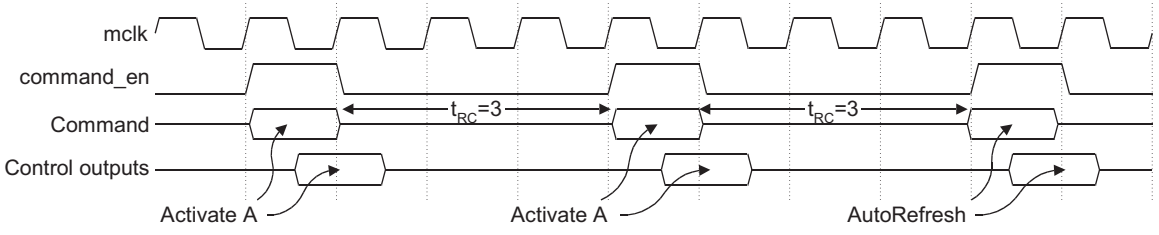


Figure 2-8 Bank activate to bank activate or auto-refresh command timing, t_{RC}

Figure 2-9 on page 2-16 shows the bank activate to different bank activate for a memory timing.

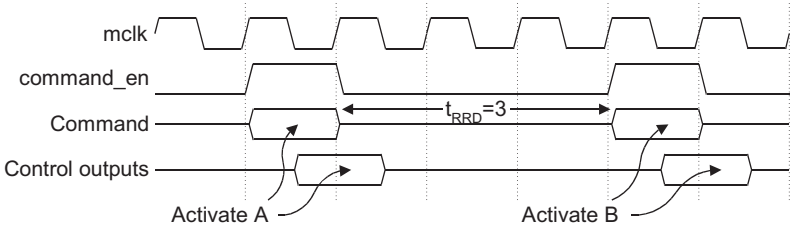


Figure 2-9 Bank activate to different bank activate for a memory timing, t_{RRD}

Figure 2-10 shows the precharge to command and auto-refresh timing.

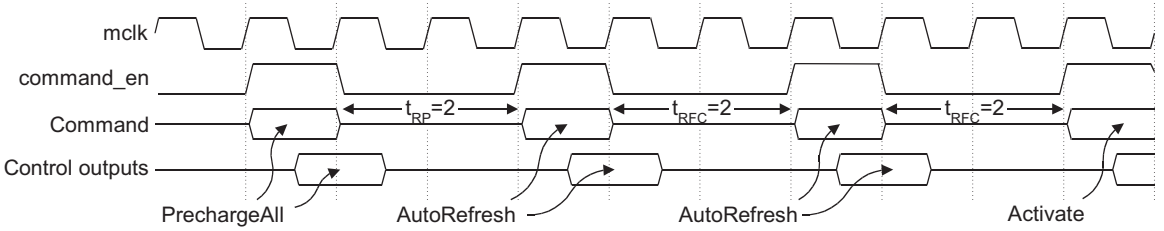


Figure 2-10 Precharge to command and auto-refresh timing, t_{RP} and t_{RFC}

Figure 2-11 shows activate to precharge, and precharge to precharge timing.

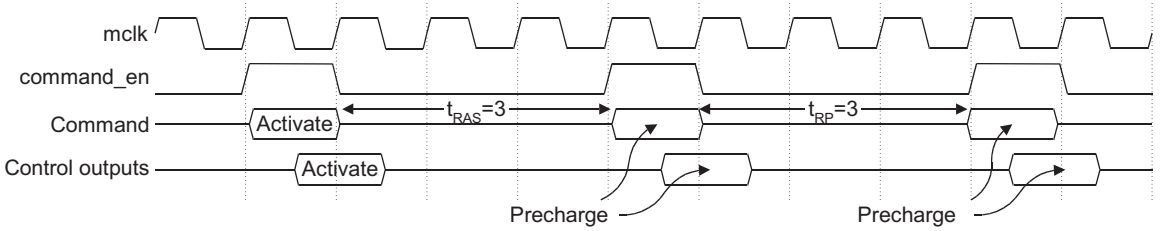


Figure 2-11 Activate to precharge, and precharge to precharge timing, t_{RAS} and t_{RP}

Figure 2-12 on page 2-17 shows mode register write to command timing.

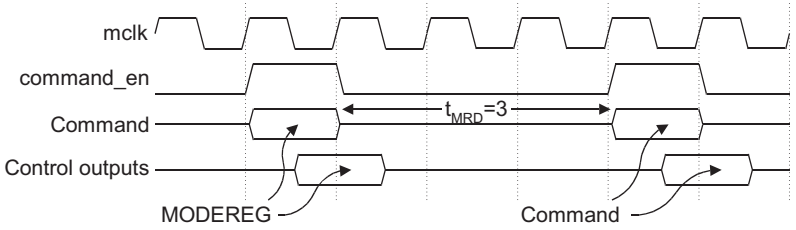


Figure 2-12 Mode register write to command timing, t_{MRD}

Figure 2-13 shows self-refresh entry and exit timing.

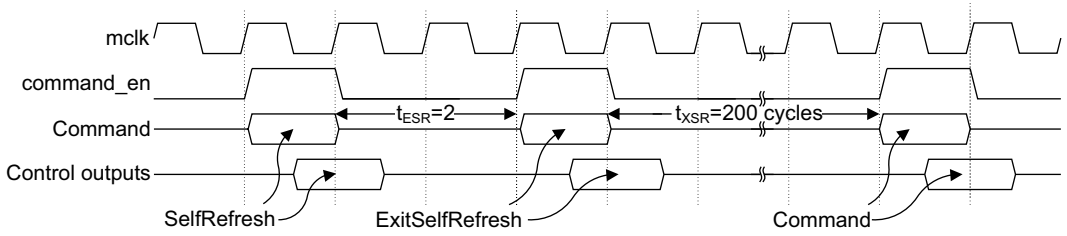


Figure 2-13 Self-refresh entry and exit timing, t_{ESR} and t_{XSR}

Figure 2-14 shows power-down entry and exit timing.

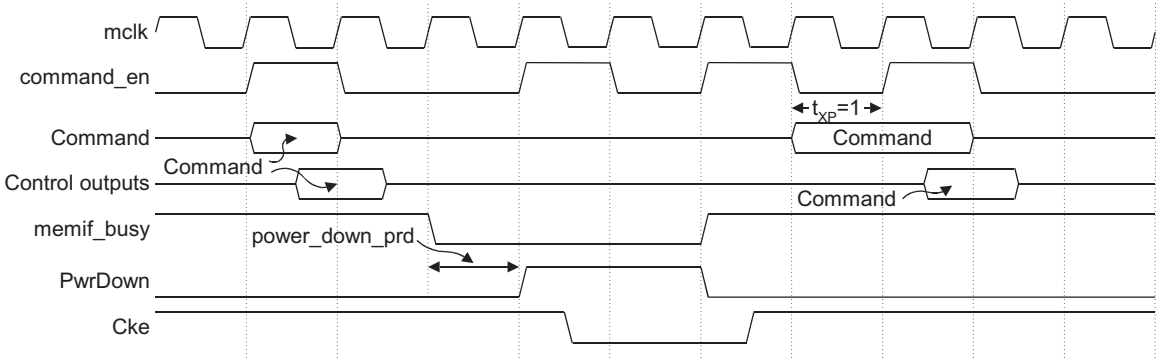


Figure 2-14 Power-down entry and exit timing, t_{XP}

The pwr_down_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. **cke** is asserted when the command FIFO is not empty.

Figure 2-15 shows the turnaround time, t_{WTR}, for the memory interface to output a Write command followed immediately by a Read command.

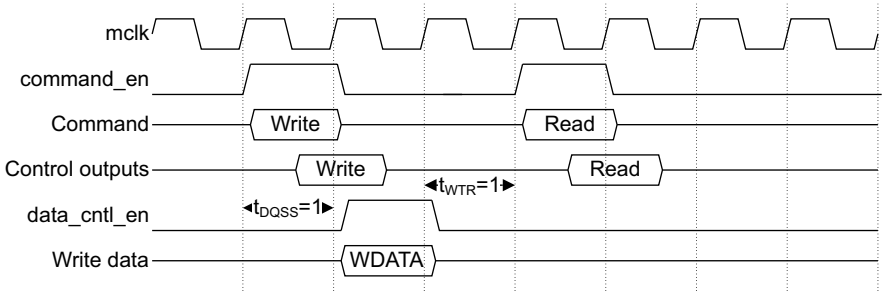


Figure 2-15 Data output timing, t_{WTR}

Figure 2-16 shows the relationship between memory interface outputting the Write command and the WDATA when t_{DQSS} is set to 1. It also highlights the t_{WR} minimum time between a Write and a Precharge command.

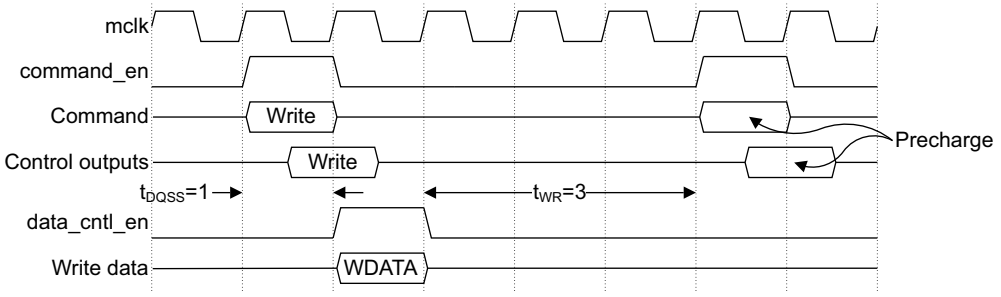


Figure 2-16 Data output timing, t_{DQSS} = 1

Figure 2-17 on page 2-19 shows the timing relationship between the Read command being output from the memory interface and the RDATA being returned to the memory interface from the pad interface.

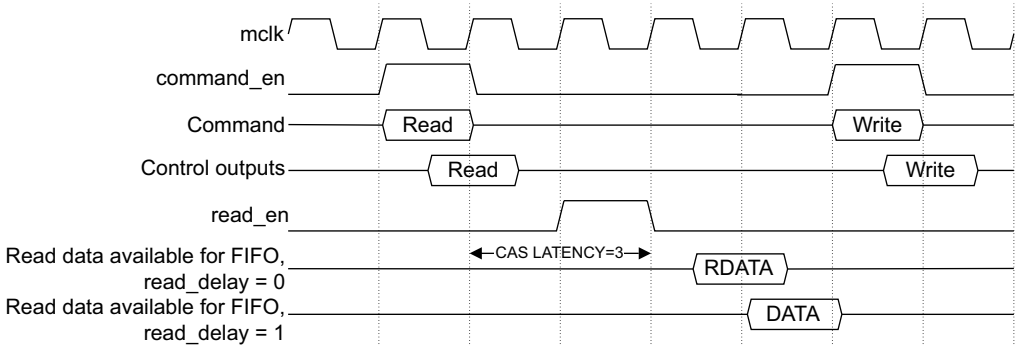


Figure 2-17 Data input timing

Note

The SDR configuration requires read_delay set to zero.

2.7 Pad interface

The pad interface is a replaceable block depending on the type of memory you are connecting. It provides a flip-flop for each external signal.

Figure 2-18 shows the pad interface external connections.

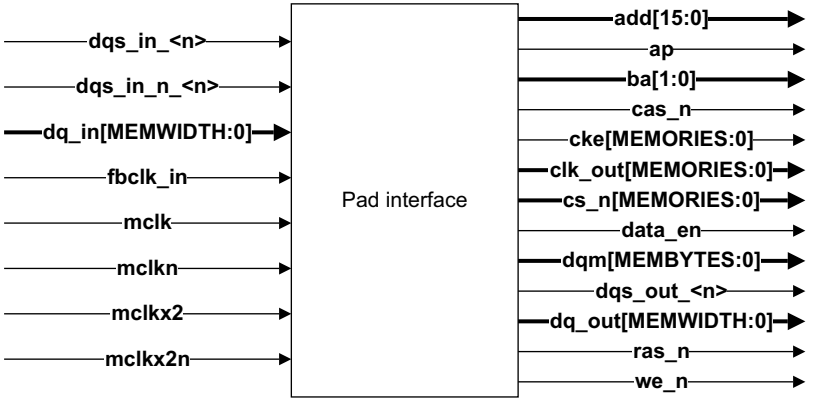


Figure 2-18 Pad interface external connections

2.7.1 Pad interface to external memory devices

It is possible that you do not require all of the signals for the external memory devices. This depends on the memory type that the pad interface block supports.

The pad interface block registers the command signals with clocks that enable the external memory device timing to be met.

To support a PrimeCell EBI (PL220), the **ap** precharge bit signal is also an external signal to the memory controller. Having **ap** separate to the address bus means that `PRECHARGEALL` commands to dynamic memory can be carried out when the EBI has granted the external memory interface to another memory controller.

It is expected, for DDR memory devices, that a *Delay-Locked Loop* (DLL) is required to delay the **dqs** signals coming back from the memories with respect to the dq data bus. The standard delay for the DQS signals is a quarter clock period of **mclk**. The DLL is not included in the memory controller.

If required, you can replace or modify the pad interface for additional optimization for a particular memory type or target library, or to use a hard macro.

If the pad interface is modified or replaced, it is important that the relative timings of the control output signals enabled by **command_en** and **data_cntl_en** signals are maintained to ensure the timings carried out in the memory interface block are still correct at the external memory bus interface. The **read_en** signal is always asserted one **mclk** period before the expected read data. Therefore, the timing of **read_en** changes as cas latency is changed using the APB interface.

When read commands are being executed, the data is clock-enabled into a FIFO clocked by **mclk** a set time from the command being clocked-out. See Figure 2-17 on page 2-19. This delay is dependent on the current cas_latency and read_delay programmed.

2.8 QoS interface

The QoS interface consists of 16 discrete I/O signals. Each I/O is associated with one of the id_<n>_cfg Registers and is referred to as the QoS override function, see also *Quality of Service* on page 2-31. The selection of the id_<n>_cfg Register is determined from four of the **ARID** bits as *Quality of Service* on page 2-31 describes.

2.9 EBI interface

The interface provides a mechanism for sharing the memory address and data buses with another memory controller, see the *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual*.

2.10 Controller management operations

The memory controller current state of operation is tracked by using a Finite State Machine. The states are shown in Figure 2-19. You can read the current status of the memory controller by reading the memc_status Register, see *Memory Controller Status Register* on page 3-7.

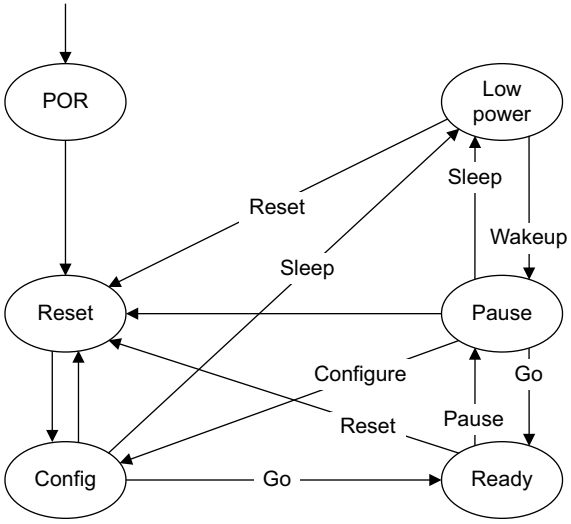


Figure 2-19 aclk domain state diagram

In Figure 2-19 non-state moving transitions are omitted for clarity.

The FSM is navigated either by accepting APB direct commands or through the AXI Low Power Interface.

If an APB command is received, that is illegal to carry out from the current state, then it is ignored and the FSM stays in the same state.

The functionality of the memory controller is controlled by the current status of the FSM.

- All the registers are available for writes or reads when the FSM is in the Config or Low-power state.
- When in the Config state or Low-power state, no auto-refresh commands are generated. When the Low-power state is entered, the SDRAM memories are put into self-refresh mode.

- When in the Ready state, not all registers are made available see Programmers Model section.
- You can achieve the Pause state through the APB interface using one of two commands in the memc_cmd Register, these are:
  - — Pause command.

    When a Pause is requested, then the Pause state is only entered when the memory controller is idle.
  - — Active_Pause command.

    When an Active_Pause is requested then the Pause state is entered when the memory interface is idle but there might still be outstanding transactions in the arbiter queue.

Note

No auto refresh commands are generated when in the Config state. If you are changing register values, it is necessary to enter the Low-power state, because this removes the risk of the memory maximum refresh time being exceeded.

The memory controller management function can issue commands to the memory interface 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 3-10. They initialize the SDRAM. The legal commands that the memory manager uses are:

- NOP
- PRECHARGEALL
- AUTOREFRESH
- MODEREG
- EXTENDED MODEREG
- DEEP POWER DOWN.

Commands from the status FSM

You can traverse the memory manager status FSM by writing to the memc_cmd Register. You can only traverse the status FSM states when the memory controller 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 memory controller is active. When you issue the command, any memory accesses that have not been arbitrated remain in the arbiter until the FSM receives a `Go` command.

Refresh commands

The refresh FSM can issue commands to the arbiter to refresh the SDRAM chips. The refresh counter is clocked by the memory clock to enable the frequency of the memory controller to be scaled without affecting the refresh rate. The refresh rate period is programmable using the refresh_prd Register. See *Refresh Period Register* on page 3-14. The value of this register is the count value in **mclk** cycles.

When the refresh counter wraps around zero, an individual auto-refresh sequence is requested for each external chip in turn.

You can prevent Refresh commands from being generated by using the active-chips bits in the memory_cfg Register.

Note

Refreshes are masked from the most significant chip number downwards.

These management commands are arbitrated with data commands.

2.11 Initialization

Before you can use the memory controller operationally to access external memory, you must:

- set up the memory controller configuration registers
- initialize the external memory devices.

You might not have to configure all the memory controller registers because some might power-up to the correct value. See Chapter 3 *Programmer's Model*. For completeness, Table 2-2 includes all register values.

Note

You might create a deadlock situation if the memory controller AXI port is accessed by a master before that master has configured the memory controller using the APB port. A master that cannot access the APB port but accesses the AXI port before the memory controller has been configured is held off until another master configures the memory controller.

2.11.1 Example setup

Table 2-2 lists an example MDDR setup.

Table 2-2 Example MDDR setup

Register address	Write data	Description
0x80000014	0x00000006	Set cas_latency to 3
0x80000018	0x00000001	Set t_dqss to 1
0x8000001C	0x00000002	Set t_mrd to 2
0x80000020	0x00000007	Set t_ras to 7
0x80000024	0x0000000B	Set t_rc to 11
0x80000028	0x00000015	Set t_rcd to 5 and schedule_rcd to 2
0x8000002C	0x000001F2	Set t_rfc to 18 and schedule_rfc to 15
0x80000030	0x00000015	Set t_rp to 5 and schedule_rp to 2
0x80000034	0x00000002	Set t_rrd to 2
0x80000038	0x00000003	Set t_wr to 3

Table 2-2 Example MDDR setup (continued)

Register address	Write data	Description
0x8000003C	0x00000002	Set t_wtr to 2
0x80000040	0x00000001	Set t_xp to 1
0x80000044	0x0000000A	Set t_xsr to 10
0x80000048	0x00000014	Set t_esr to 20
0x8000000C	0x000D0020	Set memory configuration[a]
0x80000010	0x00000A60	Set auto refresh period to be every 2656 **mclk** periods
0x80000200	0x000000FF	Set chip select for chip 0 to be 0x00XXXXXX, rbc configuration
0x80000204	0x000022FF	Set chip select for chip 1 to be 0x22XXXXXX, rbc configuration
0x80000208	0x000055FF	Set chip select for chip 2 to be 0x55XXXXXX, rbc configuration
0x8000020C	0x00007FFF	Set chip select for chip 3 to be 0x7FXXXXXX, rbc configuration
0x80000008	0x000C0000	Carry out chip0 Nop command
0x80000008	0x00000000	Carry out chip0 Prechargeall command
0x80000008	0x00040000	Carry out chip0 Autorefresh command
0x80000008	0x00040000	Carry out chip0 Autorefresh command
0x80000008	0x00080032	Carry out chip0 Mode Reg command 0x32 mapped to low add bits
0x80000008	0x001C0000	Carry out chip1 Nop command
0x80000008	0x00100000	Carry out chip1 Prechargeall command
0x80000008	0x00140000	Carry out chip1 Autorefresh command
0x80000008	0x00140000	Carry out chip1 Autorefresh command
0x80000008	0x00180032	Carry out chip1 Mode Reg command 0x32 mapped to low add bits
0x80000008	0x002C0000	Carry out chip2 Nop command
0x80000008	0x00200000	Carry out chip2 Prechargeall command
0x80000008	0x00240000	Carry out chip2 Autorefresh command
0x80000008	0x00240000	Carry out chip2 Autorefresh command

Table 2-2 Example MDDR setup (continued)

Register address	Write data	Description
0x80000008	0x00280032	Carry out chip2 Mode Reg command 0x32 mapped to low add bits
0x80000008	0x003C0000	Carry out chip3 Nop command
0x80000008	0x00300000	Carry out chip3 Prechargeall command
0x80000008	0x00340000	Carry out chip3 Autorefresh command
0x80000008	0x00340000	Carry out chip3 Autorefresh command
0x80000008	0x00380032	Carry out chip3 Mode Reg command 0x32 mapped to low add bits
0x80000004	0x00000000	Change memory controller state to Ready

a. The memory is configured as follows:
- eight column bits, 11 row bits
- precharge all bit is shared with A10
- power-down period of 0
- auto power-down is disabled
- dynamic clock stopping is disabled
- memory burst size of 4
- **ARID[5:2]** bits to be used for QoS
- force precharge is disabled
- auto self-refresh is disabled.

2.12 Data operations

All data operations are carried out through the AXI interface.

The number of outstanding AXI transactions that can be processed is a configurable option. Each outstanding transaction is referred to as:

- arbiter queue entries, or
- entries.

An entry can be created for one of two functions:

- data entries, as the result of a AXI data transactions
- management entries, as a result of management functions.

If there is a co-incident data entry and management entry request, the management entry takes priority and delays the data entry by one clock cycle.

Entries are arbitrated with an algorithm that optimizes the efficiency of the external data bus. You can modify the algorithm to meet any programmed QoS requirement.

To achieve optimum memory bus efficiency entries might be arbitrated out of order from their arrival time. Entries that cannot be arbitrated because of hazards are removed from the algorithm until the hazard is cleared.

Each arbiter queue entry contains the transaction size remaining. This is decremented each time a memory burst of data for that entry is arbitrated until the entry is complete. When all the data for an entry has been transferred across the memory interface the entry is deleted and the correct AXI response is sent.

An arbiter queue entry might not be arbitrated continuously. If a QoS event occurs then the highest priority entry changes.

The following subsections describe:

- *Hazard detection*
- *Quality of Service* on page 2-31.

2.12.1 Hazard detection

There are two types of hazard:

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.

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 *Read-After-Write* (RAW) or *Write-After-Rea*d (WAR) hazard checks in the PL340 memory controller. If an AXI master requires ordering between reads and writes to certain memory locations, it must wait for a buffered write response before issuing a read from a location it has written to (RAW). Also, it must wait for read data before issuing a write to a location it has read from (WAR).

Write transactions are also excluded from the arbitration algorithm until either:

- a full memory burst worth of data is received
- the write data burst completes
- the write data FIFO becomes full, or
- write data with a different ID is received.

2.12.2 Quality of Service

QoS is defined for the PL340 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 the arbiter receives it. No QoS exists for write accesses. There are two forms of QoS tracking:

- qos_max time-out
- qos_min time-out.

In addition, the qos_override function can dynamically force a qos_min flag for an entry.

The allocation of QoS functionality is determined by the ARID of the entry compared with a 4-bit selection mask defined by the qos_master_bits in the memory_cfg Register. The 4-bits selected can be any 4 contiguous bits from up to 8 ARID bits, that is, [3:0], [4:1], [5:2], [6:3], or [7:4].

The resultant 4-bit QoS selection number is compared against the qos_override ports that *QoS interface* on page 2-22 describes, and also enables 16 separate programmable QoS options, that the 16 id_<n>_cfg Registers. See *id_<n>_cfg Registers* on page 3-25.

If the QoS enable bit for the **ARID** is set in the register bank, the QoS maximum latency value is decremented every 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 *time-out*. Also, any entry in the queue with a minimum latency QoS also produces a time-out. Minimum latency time-outs have priority over maximum latency time-outs.

When an entry times out in this way it forces a time-out 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 qos_override function forces a minimum latency time-out. 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.

There is also a QoS provided for the auto-refresh commands from the memory manager. The arbiter keeps track of the number of auto-refresh commands in the arbiter queue with a simple increment-decrement counter. If the number of auto-refresh commands reaches a programmable tide mark, `max_out_refs`, a refresh time-out is signalled to the arbiter queue. This forces all of the auto-refresh queue entries to have a time-out. This time-out is sticky, and does not disappear when the number of time-outs drops back below the threshold. Instead, it remains asserted until all of the auto-refreshes have been serviced. This provides a guaranteed refresh rate in the SDRAM.

2.13 Low-power operation

The memory controller provides architectural support for low-power operation by supporting the SDRAM low-power modes of operation:

- automatic closing of rows
- active power-down
- precharge power-down
- automatic self-refresh entry
- self-refresh
- DPD.

Note

All functions have to be included in a configuration.

This section describes:

- *System low-power control*
- *Dynamic low-power mode control* on page 2-39
- *Deep power-down* on page 2-42.

2.13.1 System low-power control

The memory controller provides architectural support for low-power operation in the following ways:

- By using the memc_cmd and memc_status Registers at address offsets `0x4` and `0x0`. The memory controller can place the SDRAM into the self-refresh state under software control.
- By using the AXI low-power interface, the memory controller can place the SDRAM into the self-refresh state under hardware control.

Additionally, the memory controller microarchitecture provides additional power savings through extensive use of clock gating. This includes clock gating of the external memory clocks by selecting the stop_mem_clock bit in the mem_cfg Register.

You can also implement the memory controller with two power domains:

- APB and AXI, **aclk**
- memory, **mclk**.

See Figure 2-1 on page 2-2.

Table 2-3 lists the valid system states of the **aclk** domain FSM and the **mclk** domain FSM. It also lists the valid power, clock, and reset states in the **aclk** and **mclk** domains. Table 2-3 lists the valid transitions, and the text following it explains how to traverse the system states.

Table 2-3 Valid system states for FSMs

SDRAM		DMC								System state
		aclk FSM				mclk FSM				
V_{DD}	State	V_{DD}	Clock	Reset	State	V_{DD}	Clock	Reset	State	
0	Null	0	N/a	N/a	Null	0	N/a	N/a	Null	1
0	Null	>0	Running	No	POR	>0	Running	No	POR	2
0	Null	>0	Running	Yes	Reset	>0	Running	Yes	Reset	3
0	Null	>0	Running	No	Config	>0	Running	No	Powered_ up	4
>0	Accessible	>0	Running	No	Config	>0	Running	No	Powered_ up	5
>0	Accessible	>0	Running	No	Ready	>0	Running	No	Powered_ up	6
>0	Powered-down	>0	Running	No	Ready	>0	Running	No	Powered_ down	7
>0	Self_ refresh	>0	Running	No	Low_ power	>0	Running	No	Self_ refresh	8
>0	Self_ refresh	>0	Running	No	Low_ power	>0	Stopped	No	Self_ refresh	9
>0	Self_ refresh	>0	Stopped	No	Low_ power	>0	Running	No	Self_ refresh	10
>0	Self_ refresh	>0	Stopped	No	Low_ power	>0	Stopped	No	Self_ refresh	11
>0	Self_ refresh	0	N/a	N/a	Null	>0	Stopped	No	Self_ refresh	12
>0	Self_ refresh	0	N/a	N/a	Null	>0	Running	No	Self_ refresh	13
>0	Self_ refresh	>0	Running	No	POR	>0	Stopped	No	Self_ refresh	14
>0	Self_ refresh	>0	Running	No	POR	>0	Running	No	Self_ refresh	15

Table 2-3 Valid system states for FSMs (continued)

SDRAM		DMC								System state
		aclk FSM				mclk FSM				
V_{DD}	State	V_{DD}	Clock	Reset	State	V_{DD}	Clock	Reset	State	
>0	Self_ refresh	>0	Running	Yes	Reset	>0	Stopped	No	Self_ refresh	16
>0	Self_ refresh	>0	Running	Yes	Reset	>0	Running	No	Self_ refresh	17
>0	Self_refresh	>0	Stopped	No	Ready	>0	Stopped	No	Self_refresh	18

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

6, 7, 8, 10, 9, 11, 13, 12.

However, states 8-11 are similar and the recommendation is to use state 11 from this group if clock-stopping techniques are available. Similarly, states 12 & 13 are similar and the recommendation is to use state 12 from this pair. Table 2-4 lists a recommended set of power states.

Table 2-4 Recommended power states

System state	Power name
6	Running
7	Auto power-down
11	Shallow self-refresh or auto self-refresh
12	Deep self-refresh
18	Auto self-refresh

Figure 2-20 on page 2-36 highlights these states and arcs.

Note

States 1-5, 9, 14, and 16 are only used as transitional states.

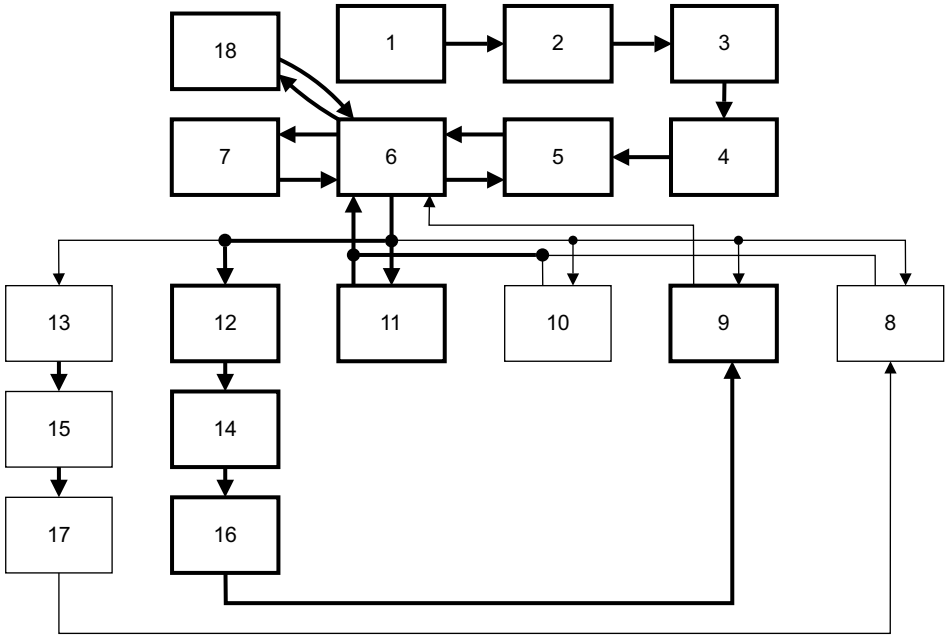


Figure 2-20 System state transitions

State transitions are as follows:

Arc 1 to 2 Apply power to all memory controller power domains, and ensure that **aclk** and **mclk** are running.

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

Arc 3 to 4 Deassert reset in both the **aclk** reset domain and the **mclk** reset domain.

Arc 4 to 5 Apply power to the SDRAM power domain.

Arc 5 to 6 You must:

1. Write to all of the memory timing parameters, address offsets `0x14` to `0x44`.
2. Write to the memory_cfg and refresh_prd Registers, address offsets `0xC` and `0x10`.
3. Initialize the memory, using the direct_cmd Register, offset `0x8`, with the sequence of commands specified by the memory vendor. When you have sent these commands to the memory, you can write to the memc_cmd Register, offset `0x4` with the Go command, `0x0`.

4. Poll the memc_status Register until the value of `0x1` is returned, Ready, signifying that the memory controller is ready to accept AXI accesses to the SDRAM.

Arc 6 to 5 If you want to reconfigure either the memory controller or SDRAM, you must first write to the memc_cmd Register, offset `0x4`, with the Pause command, `0x3`, and poll the memc_status Register until the value of `0x2` is returned, Paused. Then you can write to the memc_cmd Register with the Configure command, `0x4` and poll the memc_status Register until the value of `0x0` is returned, Config.

Arc 6 to 7 If auto_power_down is set in the memory_cfg Register, see *Memory Configuration Register* on page 3-12, then this arc is automatically taken when the SDRAM has been idle for power_down_prd **mclk** cycles.

Arc 7 to 6 When an SDRAM access command has been received in the **mclk** domain, this arc is taken.

Arc 6 to 8 You can take this arc under either hardware or software control:

- To take this arc under software control:
  1. Issue the Pause command, or archive the Pause command.
  2. Poll for the Paused state.
  3. Issue the Sleep command.
- To take this arc under hardware control, use the AXI low-power interface to request a Low-power state.

Arc 6 to 9 The same as arc 6 to 8, but additionally stop the **mclk** domain clock.

Arc 6 to 10 The same as arc 6 to 8, but additionally stop the **aclk** domain clock.

Arc 6 to 11 The same as arc 6 to 8, but additionally stop both the **mclk** and the **aclk** domain clocks.

Arc 6 to 12 The same as arc 6 to 8, but additionally stop the **mclk** domain clock and remove power from the **aclk** power domain. This can only be done if the memory controller implementation has separate power domains for **aclk** and **mclk**.

Arc 6 to 13 The same as arc 6 to 8, but additionally remove power from the **aclk** power domain. This can only be done if the memory controller implementation has separate power domains for **aclk** and **mclk**.

Arc 8 to 6 You can take this arc under either hardware or software control:

- To take this arc under software control:
  1. Issue the Wakeup command to 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.
- To take this arc under hardware control, use the AXI low-power interface to bring the memory controller out of a low-power state.

Arc 9 to 6 The same as arc 8 to 6, but you must first start the **mclk** domain clock.

Arc 10 to 6 The same as arc 8 to 6, but you must first start the **aclk** domain clock.

Arc 11 to 6 The same as arc 8 to 6, but you must first start both the **aclk** and **mclk** domain clocks.

Arc 12 to 14 Apply power to the **aclk** power domain.

Arc 14 to 16 Assert reset to the **aclk** reset domain.

Arc 16 to 9 De-assert reset to the **aclk** reset domain.

Arc 13 to 15 Apply power to the **aclk** power domain.

Arc 15 to 17 Assert reset to the **aclk** reset domain.

Arc 17 to 8 De-assert reset to the **aclk** reset domain.

Arc 6 to 18 If auto_power_down is set in the memory_cfg Register, see *Memory Configuration Register* on page 3-12, then this arc is automatically taken when the SDRAM has been idle for `power_down_prd` **mclk** cycles. Also requires: `fp_line` << `power_down_prd`, `fp_enable` and `sr_enable`.

Arc 18 to 6 When an SDRAM access command has been received in the **mclk** domain, this arc is taken.

Note

When power is applied to the **aclk** domain, when leaving state 1, the memory controller status FSM moves to the Config state. When power is applied to the **aclk** domain, when leaving states 12 or 13, the memory controller states FSM moves to the Low-power state.

2.13.2 Dynamic low-power mode control

Dynamic low-power mode control operates when the memory controller is in the Ready state.

The functionality that Table 2-5 lists is dependant on whether the memory controller is configured to have a single global **cke** or a **cke** per memory device. The functionality works for each **cke** pin, when using a:

global cke All memory devices must be idle.

local cke A single memory device can be entered into a low-power mode of operation.

The functionality listed in Table 2-5 is configurable and programmable. A memory controller configuration requires the functionality to be included before it can be enabled through the APB interface.

Note

When auto self-refresh entry is configured, a 10-bit prescalar is also configured to increase the time of the `auto_power_prd`. Auto self-refresh entry cannot work without the `auto_power_down` functionality being enabled and the force precharge functionality being enabled.

Table 2-5 lists the dynamic low-power modes operation.

Table 2-5 Dynamic low-power modes operation

Auto power-down	Force precharge	Auto self-refresh entry	Operation
0	0	0	No power saving
0	0	1	No power saving
0	1	0	Force precharge after fp_time
0	1	1	Force precharge after fp_time
1	0	0	Auto-power-down after power_down_prd
1	0	1	Auto-power-down after power_down_prd
1	1	0	Force precharge after fp_time plus Auto-power-down after power_down_prd
1	1	1	Force precharge after fp_time plus self-refresh entry after power_down_prd

Auto power-down logic negates **cke** for a memory chip putting the device into either active or precharge power-down mode if a device has been idle for `power_down_prd` time depending on whether the device had any open rows. The **mclk** clock decrements the power-down counters. The programmed `power_down_prd` time must always be greater than the programmed cas latency.

Force precharge logic automatically generates a precharge for an idle activated bank. If a bank has been activated and has executed a data access then subsequently, if no more data accesses are executed for `fp_time`, then a force precharge is generated to close that idle bank. The **aclk** clock decrements the force precharge counters.

Auto self-refresh logic operates the same as the auto power-down logic but generates a self-refresh entry for a memory device command instead of negating **cke**.

Figure 2-21 shows the time after completion of a command to a memory chip until the memory controller 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.

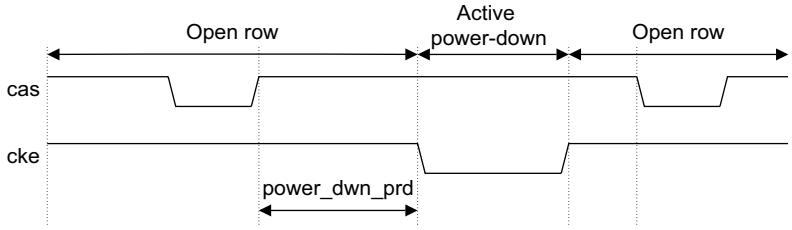


Figure 2-21 Auto-power-down

Figure 2-22 on page 2-41 shows the time after completion of a command to a memory chip until the memory controller places that chip into power-down mode. When `fp_enable` is set with `fp_time` set to zero then the equivalent functionality of auto-precharge commands is achieved.

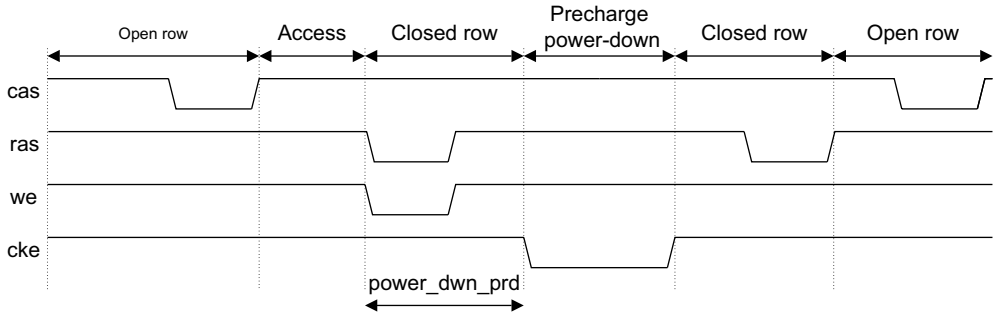


Figure 2-22 Force precharge with zero force precharge time

Figure 2-23 shows the time after completion of a command to a memory chip until the memory controller places that chip into precharge power-down mode. To ensure precharge power-down mode for every bank, you must set `fp_time` to less than `power_down_prd` - 3 for synchronous 1:1 clocking and scaled accordingly for different clocking modes. For example if **mclk** is running 2 times slower than **aclk** then `fp_time` must be:

((`fp_time` x 2) + 3) < `power_down_prd`

When running asynchronously the `fp_time` must be scaled to ensure it must always be less than the `power_down_prd` - 3. This ensures that a precharge always occurs before the **cke** pin is negated.

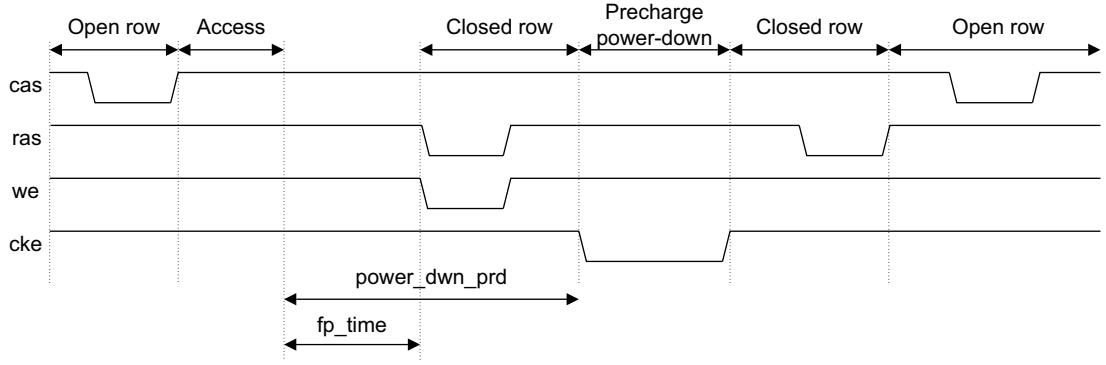


Figure 2-23 Force precharge after pd time

Figure 2-24 on page 2-42 shows the time after completion of a command to a memory chip, until the memory controller places that chip into self-refresh mode. Table 2-5 on page 2-39 shows the memory controller can only put a chip into self-refresh mode if the force precharge logic and auto self-refresh logic is configured and enabled. This

guarantees that if a self-refresh command is generated all the banks of a chip have been previously precharged. When the auto self-refresh command logic is configured a 10-bit prescalar for each `power_down_prd` counter is generated. If the prescale value is programmed to zero then the prescalar does not affect the power-down counter.

A prescalar is auto-generated because for some memory types there is a relatively long time for a memory chip to exit self-refresh mode which stalls the command FIFO. Therefore, because the penalty for exiting self-refresh mode is large, you can program a chip select to be idle for a much longer time before entering self-refresh mode when compared to the other power-down modes.

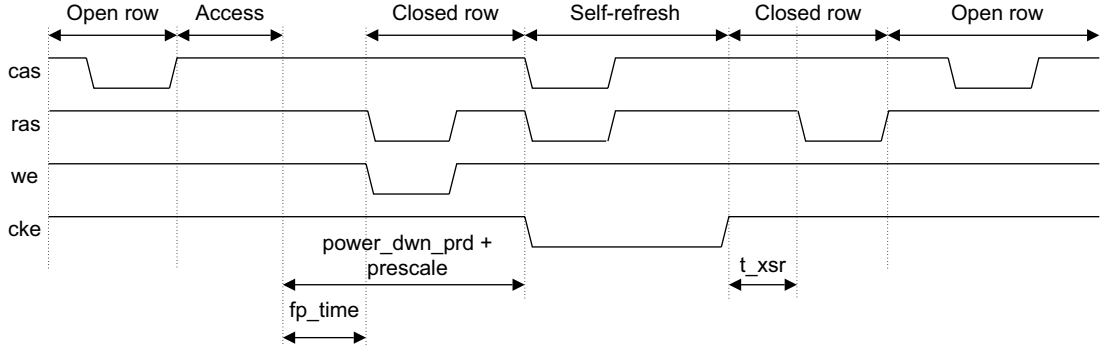


Figure 2-24 Auto self-refresh entry

2.13.3 Deep power-down

You can only achieve *Deep Power Down* (DPD) for individual chips if the memory controller configuration has local **cke** set, that is, a **cke** pin for each memory device.

For a memory controller configuration with a global **cke** it is only possible to enter all the memory devices into the DPD state simultaneously.

To ensure that no refreshes are generated for a chip select that has been put into DPD mode, the active_chips bits must be reduced in line with the DPD commands. This means that DPD mode can only be entered from the most significant chip select of a configuration downwards.

The system architect must ensure that no data transactions are sent to a chip select that has been entered into DPD mode.

The flow for entering power-down mode is as follows:

1. Enter Pause state.
2. Enter Low-power state.

3. Write a direct command to the highest chip select number configured with a PRECHARGEALL command, so as to only enter DPD with all chips precharged.
4. Write a direct command to the highest chip select number configured with a DPD command.
5. Write direct command to the next highest chip select number if required.
6. Write to the active_chips bits of the memory_configuration register to disable refreshes for the power-down chip selects.

To remove a chip select from DPD is the reverse of the above sequence, excluding the PRECHARGEALL command.

Note

A chip select must have the active_chips bits set before providing the NOP direct command and then carrying out the memory initialization sequence.

All the registers are available for writes or reads when the FSM is in the Config or Low-power state.

Chapter 3
Programmer's Model

This chapter describes the memory controller registers and provides information for programming the device. It contains the following sections:

3.1 About the programmer's model

Figure 3-1 shows the memory controller register map.

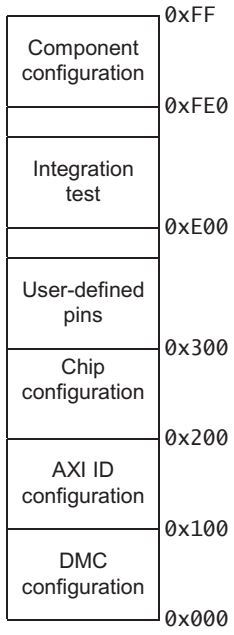


Figure 3-1 Register map

3.2 Register summary

Figure 3-2 shows the memory controller configuration register map.

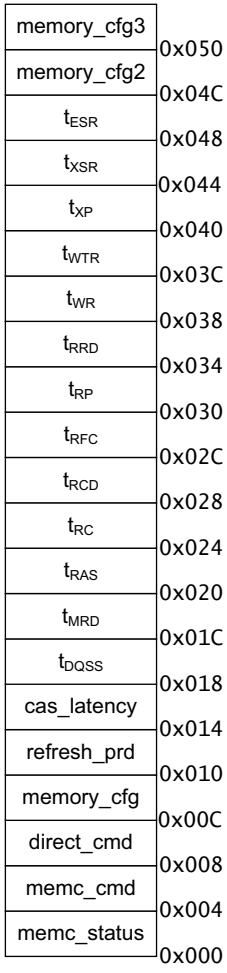


Figure 3-2 Memory controller configuration register map

Figure 3-3 on page 3-4 shows the AXI ID configuration registers map.

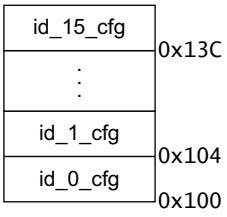


Figure 3-3 AXI ID configuration register map

Figure 3-4 shows the chip configuration register map.

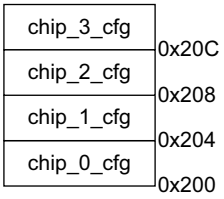


Figure 3-4 Chip configuration registers map

Figure 3-5 shows the identification register map.

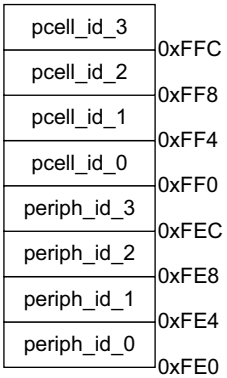


Figure 3-5 Identification register map

Table 3-1 on page 3-5 lists the memory controller registers.

Table 3-1 Memory controller register summary

Name	Base offset	Type	Reset value	Description
memc_status	0x000	RO	-[a]	*Memory Controller Status Register* on page 3-7
memc_cmd	0x004	WO	-	*Memory Controller Command Register* on page 3-9
direct_cmd	0x008	WO	-	*Direct Command Register* on page 3-10
memory_cfg	0x00C	RW	0x00010020	*Memory Configuration Register* on page 3-12
refresh_prd	0x010	RW	0x00000A60	*Refresh Period Register* on page 3-14
cas_latency	0x014	RW	0x00000006	*CAS Latency Register* on page 3-15
t_dqss	0x018	RW	0x00000001	*t_dqss Register* on page 3-16
t_mrd	0x01C	RW	0x00000002	*t_mrd Register* on page 3-16
t_ras	0x020	RW	0x00000007	*t_ras Register* on page 3-17
t_rc	0x024	RW	0x0000000B	*t_rc Register* on page 3-17
t_rcd	0x028	RW	0x0000001D	*t_rcd Register* on page 3-18
t_rfc	0x02C	RW	0x00000212	*t_rfc Register* on page 3-19
t_rp	0x030	RW	0x0000001D	*t_rp Register* on page 3-19
t_rrd	0x034	RW	0x00000002	*t_rrd Register* on page 3-20
t_wr	0x038	RW	0x00000003	*t_wr Register* on page 3-20
t_wtr	0x03C	RW	0x00000002	*t_wtr Register* on page 3-21
t_xp	0x040	RW	0x00000001	*t_xp Register* on page 3-22
t_xsr	0x044	RW	0x0000000A	*t_xsr Register* on page 3-22
t_esr	0x048	RW	0x00000014	*t_esr Register* on page 3-23
memory_cfg2	0x04C	RW	-[b]	*memory_cfg2 Register* on page 3-23
memory_cfg3	0x050	RW	0x00000006	*memory_cfg3 Register* on page 3-25
-	0x054-0x0FC	-	-	-
id_n_cfg	0x100	RW	0x00000000	*id_<n>_cfg Registers* on page 3-25
chip_n_cfg	0x200	RW	0x0000FF00	*chip_<n>_cfg Registers* on page 3-26
user_status	0x300	RO	-	*user_status Register* on page 3-27

Table 3-1 Memory controller register summary (continued)

Name	Base offset	Type	Reset value	Description
user_config	0x304	WO	-	*user_config Register* on page 3-27
periph_id_n	0xFE0-0xFEC	RO	-	*Peripheral Identification Registers 0-3* on page 3-28
pcell_id_n	0xFF0-0xFFC	RO	-	*PrimeCell Identification Registers 0-3* on page 3-30

a. Dependent on configuration.
b. Dependent on tie-off port values.

3.3 Register descriptions

This section describes the memory controller registers.

3.3.1 Memory Controller Status Register

The read-only memc_status Register provides information on the configuration of the memory controller, and also on the current state of the memory controller. It cannot be read in either the Reset or POR states. Figure 3-6 shows the register bit assignments.

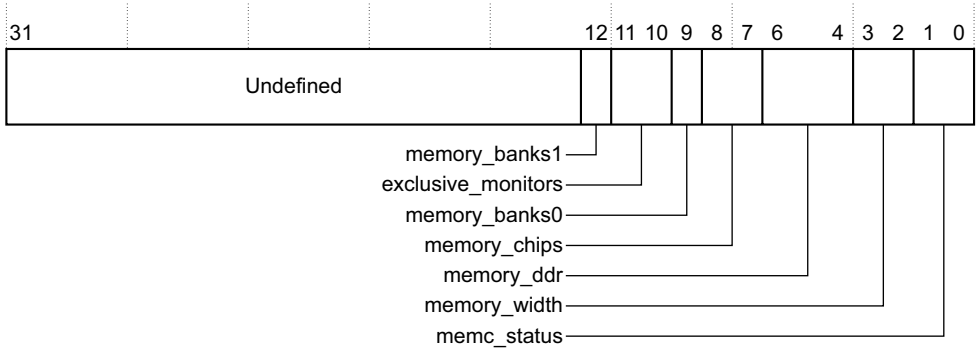


Figure 3-6 memc_status Register bit assignments

Table 3-2 lists the register bit assignments.

Table 3-2 memc_status Register bit assignments

Bits	Name	Function
[31:13]	-	Read undefined.
[12:]	memory_banks1	This returns part of the definition of the number of banks that the PL340 supports on each chip.[a]
[11:10]	exclusive_monitors	Returns the number of exclusive access monitor resources implemented in the memory controller: 2'b00 = 0 monitors 2'b01 = 1 monitor 2'b10 = 2 monitors 2'b11 = 4 monitors.
[9]	memory_banks0	This returns part of the definition of the number of banks that the PL340 supports on each chip.[a]

Table 3-2 memc_status Register bit assignments (continued)

Bits	Name	Function
[8:7]	memory_chips	Returns the number of different chip selects that the memory controller supports: 2'b00 = 1 chip 2'b01 = 2 chips 2'b10 = 3 chips 2'b11 = 4 chips.
[6:4]	memory_type	Returns the SDRAM that the memory controller supports: 3'b000 = SDR SDRAM 3'b001 = DDR SDRAM 3'b011 = Mobile DDR SDRAM 3'b010 = eDRAM 3'b1xx = Reserved. If Mobile DDR SDRAM or SDR SDRAM or an eDRAM is supported, the cas_half_cycle bit at address offset `0x14` is ignored.
[3:2]	memory_width	Returns the width of the external memory: 2'b00 = 16-bit 2'b01 = 32-bit 2'b10 = 64-bit 2'b11 = Reserved.
[1:0]	memc_status	Returns the state of the memory controller: 2'b00 = Config 2'b01 = Ready 2'b10 = Paused 2'b11 = Low_power.

a. See Table 3-3 on page 3-9 for memory banks chip configuration information.

Table 3-3 lists the memory banks chip configurations.

Table 3-3 Memory banks chip configuration

Memorybanks1 and Memorybanks0	Banks per memory chip
0	4
1	2[a]
2	Reserved
3	Reserved

a. Two memory banks per chip is applicable for only eDRAM configurations.

3.3.2 Memory Controller Command Register

The write-only memc_cmd Register enables the memory controller to be traversed. The command register controls the programmer's view FSM. By writing to this register, the FSM can be traversed. If a new command is received to change state, and a previous command to change state has not completed, the **pready** signal is held LOW until the new command can be carried out.

Figure 3-7 shows the register bit assignments.

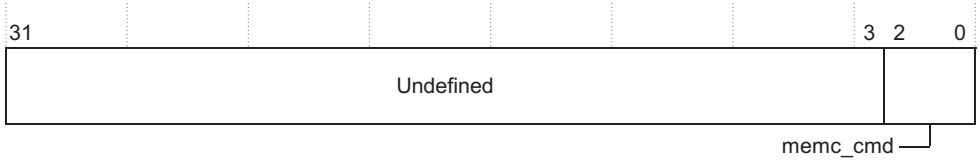


Figure 3-7 memc_cmd Register bit assignments

Table 3-4 lists the register bit assignments.

Table 3-4 memc_cmd Register bit assignments

Bits	Name	Function
[31:3]	-	Undefined. Write as zero.
[2:0]	memc_cmd	Changes the state of the memory controller: 3'b000 = Go 3'b001 = Sleep 3'b010 = Wakeup 3'b011 = Pause 3'b100 = Configure 3'b111 = Active_Pause.

Note

`Active_Pause` puts the memory controller into the Paused state without draining the arbiter queue. This enables you to enter low-power mode to change configuration settings such as memory frequency or timing register values without requiring coordination between masters in a multi-master system. If the memory controller is put into low-power mode after using the `Active_Pause` command, you must not remove power from the memory controller because this results in data loss and violation of the AXI protocol. The memory controller does not issue refreshes while in the Config state. You must use low-power mode to make register updates because this ensures that the memory is put into self-refresh rather than entering the Config state when the memory contains valid data.

3.3.3 Direct Command Register

The write-only direct_cmd Register passes commands to the external memory. The configuration of the direct_cmd Register enables you to write to any type of Mode register supported by the external memory device, and also to generate NOP, Prechargeall, and Auto-refresh commands. 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 defined in the timing registers. Therefore, if an initialization sequence requires additional delays between commands, they must be timed by the master driving the initialization sequence.

This register can only be written to in the Config or Low-power state. Figure 3-8 on page 3-11 shows the register bit assignments.

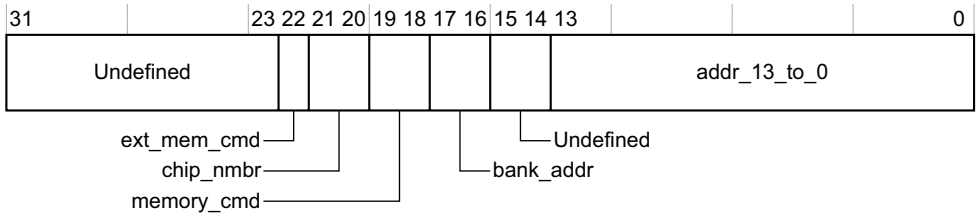


Figure 3-8 direct_cmd Register bit assignments

Table 3-5 lists the register bit assignments.

Table 3-5 direct_cmd Register bit assignments

Bits	Name	Function
[31:23]	-	Undefined, write as zero
[22]	ext_mem_cmd	Extended memory command, see note after the table
[21:20]	chip_nmbr	Bits mapped to external memory chip address bits
[19:18]	memory_cmd	Determines the command required, see note after the table
[17:16]	bank_addr	Bits mapped to external memory bank address bits when command is Modereg access
[15:14]	-	Undefined, write as zero
[13:0]	addr_13_to_0	Bits mapped to external memory address bits [13:0] when command is Modereg access

Note

Memory command encoding. This encoding uses the ext_mem_cmd bits concatenated to memory_cmd, therefore providing 3 bits.

3'b000 = Prechargeall

3'b001 = Autorefresh

3'b010 = Modereg or Extended modereg access

3'b011 = NOP

3'b100 = DPD

All other combinations are illegal and might cause undefined behavior.

A NOP command asserts all chip selects that are set as active_chips when the chip_nmbr is set to 0.

If chip_nmbr is set to 1 only **cs_n[1]** is asserted.

If chip_nmbr is set to 2 only **cs_n[2]** is asserted.

If chip_nmbr is set to 3 only **cs_n[3]** is asserted.

3.3.4 Memory Configuration Register

The read/write memory_cfg Register configures the memory. It can only be read from and written to in the Config or Low-power state. Figure 3-9 shows the register bit assignments.

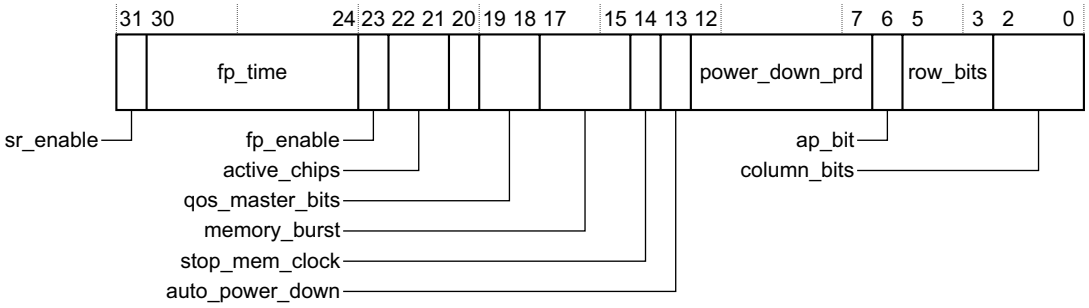


Figure 3-9 memory_cfg Register bit assignments

Table 3-6 lists the register bit assignments.

Table 3-6 memory_cfg Register bit assignments

Bits	Name	Function
[31]	sr_enable	Auto self-entry enable. Only if configured else read is undefined write as zero. See Chapter 2 *Functional Overview*.
[30:24]	fp_time	Force precharge timeout count. Only valid if configured, else read is undefined write as zero. See Chapter 2 *Functional Overview*.
[23]	fp_enable	Force precharge enable. Only valid if configured, else read undefined write as zero. See Chapter 2 *Functional Overview*.

Table 3-6 memory_cfg Register bit assignments (continued)

Bits	Name	Function
[22:21]	active_chips	Enables the refresh command generation for the number of memory 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: 2'b00 = 1 chip 2'b01 = 2 chips 2'b10 = 3 chips 2'b11 = 4 chips.
[20:18]	qos_master_bits	Encodes the four bits of the 8-bit AXI ARID that select one of the 16 QOS values: 3'b000 = ARID[3:0] 3'b001 = ARID[4:1] 3'b010 = ARID[5:2] 3'b011 = ARID[6:3] 3'b100 = ARID[7:4].
[17:15]	memory_burst	Encodes the number of data accesses that are performed to the SDRAM for each Read and Write command: 3'b000 = Burst 1 3'b001 = Burst 2 3'b010 = Burst 4 3'b011 = Burst 8 3'b100 = Burst 16. This value must also be programmed into the SDRAM mode register using the direct_cmd register at offset `0x8`, and must match it.
[14]	stop_mem_clock	When enabled, the memory clock is dynamically stopped when not performing an access to the SDRAM.
[13]	auto_power_down	When this is set, the memory interface automatically places the SDRAM into power-down state by deasserting **cke** when the command FIFO has been empty for power_down_prd memory clock cycles.
[12:7]	power_down_prd	Number of memory clock cycles for auto power-down of the SDRAM. The power_down_prd programmed must be greater than the programmed cas latency

Table 3-6 memory_cfg Register bit assignments (continued)

Bits	Name	Function
[6]	ap_bit	Encodes the position of the auto-precharge bit in the memory address: 1'b0 = address bit 10 1'b1 = address bit 8.
[5:3]	row_bits	Encodes the number of bits of the AXI address that comprise the row address: 3'b000 = 11 bits 3'b001 = 12 bits 3'b010 = 13 bits 3'b011 = 14 bits 3'b100 = 15 bits 3'b101 = 16 bits. The combination of row size, column size, BRC/RBC, and memory width must ensure that neither the MSB of the row address nor the MSB of the bank address exceed address range [27:0].
[2:0]	column_bits	Encodes the number of bits of the AXI address that comprise the column address: 3'b000 = 8 bits 3'b001 = 9 bits 3'b010 = 10 bits 3'b011 = 11 bits 3'b100 = 12 bits.

3.3.5 Refresh Period Register

The read/write refresh_prd Register sets the memory refresh period. It can only be read from and written to in the Config or Low-power state. Figure 3-10 shows the register bit assignments.

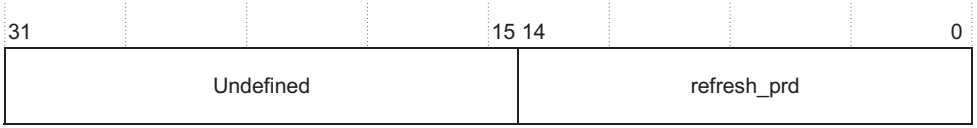


Figure 3-10 refresh_prd Register bit assignments

Table 3-7 lists the 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 memory clock cycles

3.3.6 CAS Latency Register

The read/write cas_latency Register sets the CAS latency in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-11 shows the register bit assignments. See also *Memory interface* on page 2-13.

Note

You are required to match the programmed cas latency with the cas latency written to the memory device through the direct_cmd Register.

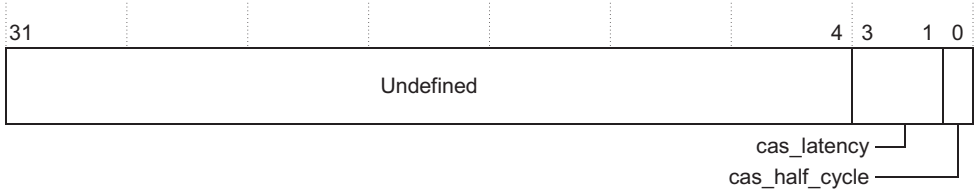


Figure 3-11 cas_latency Register bit assignments

Table 3-8 lists the register bit assignments.

Table 3-8 cas_latency Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero.
[3:1]	cas_latency	CAS latency in memory clock cycles.
[0]	cas_half_cycle	Encodes whether the CAS latency is half a memory clock cycle more than the value given in bits [3:1]: 1'b0 = Zero cycles offset to value in [3:1]. b0 is forced to 0 in MDDR and SDR mode. 1'b1 = Half cycle offset to value in [3:1].

3.3.7 t_dqss Register

The read/write t_dqss Register writes to DQS in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-12 shows the register bit assignments. See also *Memory interface* on page 2-13.

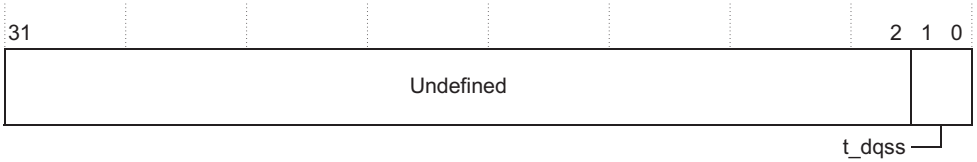


Figure 3-12 t_dqss Register bit assignments

Table 3-9 lists the register bit assignments.

Table 3-9 t_dqss Register bit assignments

Bits	Name	Function
[31:2]	-	Read undefined, write as zero
[1:0]	t_dqss	Write to DQS in memory clock cycles

3.3.8 t_mrd Register

The read/write t_mrd Register sets the mode register command time in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-13 shows the register bit assignments. See also *Memory interface* on page 2-13. See also *Memory interface* on page 2-13.

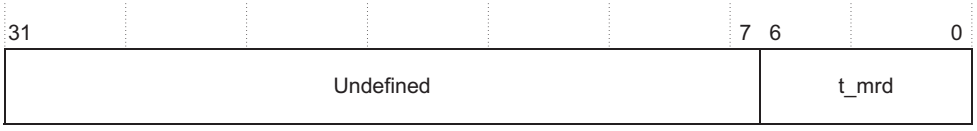


Figure 3-13 t_mrd Register bit assignments

Table 3-10 lists the 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 mode register command time in memory clock cycles

3.3.9 t_ras Register

The read/write t_ras Register sets the RAS to precharge delay in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-14 shows the register bit assignments. See also *Memory interface* on page 2-13.

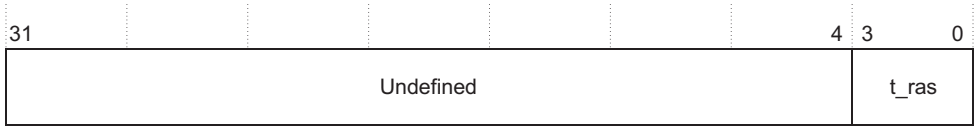


Figure 3-14 t_ras Register bit assignments

Table 3-11 lists the register bit assignments.

Table 3-11 t_ras Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero
[3:0]	t_ras	Sets RAS to precharge delay in memory clock cycles

3.3.10 t_rc Register

The read/write t_rc Register sets the Active bank x to Active bank x delay in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-15 shows the register bit assignments. See also *Memory interface* on page 2-13.

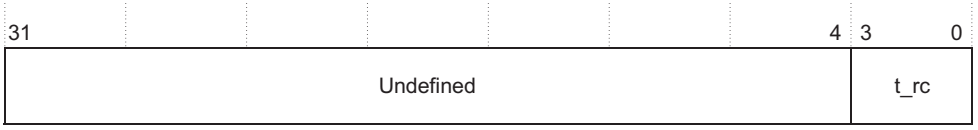


Figure 3-15 t_rc Register bit assignments

Table 3-12 lists the register bit assignments.

Table 3-12 t_rc Register bit assignments

Bits	Name	Function
[31:4]	-	Read undefined, write as zero
[3:0]	t_rc	Sets Active bank x to Active bank x delay in memory clock cycles

3.3.11 t_rcd Register

The read/write t_rcd Register sets the RAS to CAS minimum delay in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-16 shows the register bit assignments. See also *Memory interface* on page 2-13.

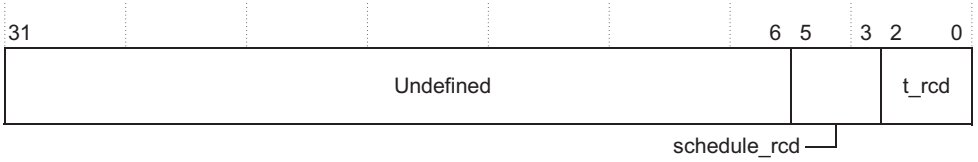


Figure 3-16 t_rcd Register bit assignments

Table 3-13 lists the register bit assignments.

Table 3-13 t_rcd Register bit assignments

Bits	Name	Function
[31:6]	-	Read undefined, write as zero.
[5:3]	schedule_rcd	Sets the RAS to CAS minimum delay in **aclk** cycles -3. See also note in *Memory interface to pad interface timing* on page 2-14.
[2:0]	t_rcd	Sets the RAS to CAS minimum delay in memory clock cycles.

3.3.12 t_rfc Register

The read/write t_rfc Register sets the auto-refresh command time in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-17 on page 3-19 shows the register bit assignments. See also *Memory interface* on page 2-13.

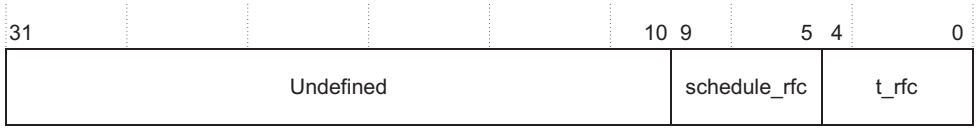


Figure 3-17 t_rfc Register bit assignments

Table 3-14 lists the register bit assignments.

Table 3-14 t_rfc Register bit assignments

Bits	Name	Function
[31:10]	-	Read undefined, write as zero.
[9:5]	schedule_rfc	Sets the autorefresh command time in **aclk** cycles -3. See also note in *Memory interface to pad interface timing* on page 2-14.
[4:0]	t_rfc	Sets the auto-refresh command time in memory clock cycles.

3.3.13 t_rp Register

The read/write t_rp Register sets the precharge to RAS delay in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-18 shows the register bit assignments. See also *Memory interface* on page 2-13.

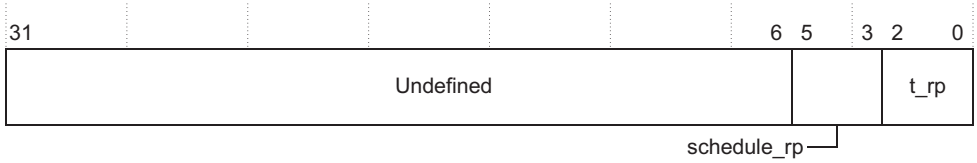


Figure 3-18 t_rp Register bit assignments

Table 3-15 lists the register bit assignments.

Table 3-15 t_rp Register bit assignments

Bits	Name	Function
[31:6]	-	Read undefined, write as zero.
[5:3]	schedule_rp	Sets the precharge to RAS delay in **aclk** cycles -3. See also note in *Memory interface to pad interface timing* on page 2-14.
[2:0]	t_rp	Sets the precharge to RAS delay in memory clock cycles.

3.3.14 t_rrd Register

The read/write t_rrd Register sets the Active bank x to Active bank y delay in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-19 shows the register bit assignments. See also *Memory interface* on page 2-13.

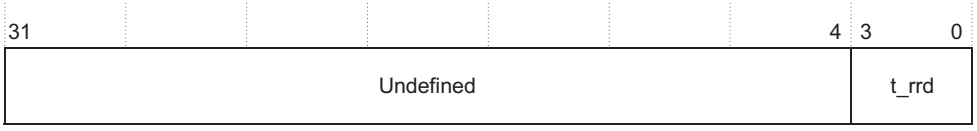


Figure 3-19 t_rrd Register bit assignments

Table 3-16 lists the 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 Active bank x to Active bank y delay in memory clock cycles

3.3.15 t_wr Register

The read/write t_wr Register sets the write to precharge delay in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-21 on page 3-21 shows the register bit assignments. See also *Memory interface* on page 2-13.

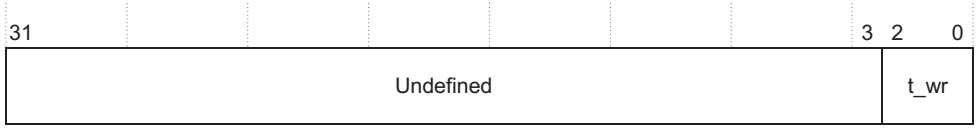


Figure 3-20 t_wr Register bit assignments

Table 3-17 lists the 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 the write to precharge delay in memory clock cycles

3.3.16 t_wtr Register

The read/write t_wtr Register sets the write to read delay in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-21 shows the register bit assignments. See also *Memory interface* on page 2-13.

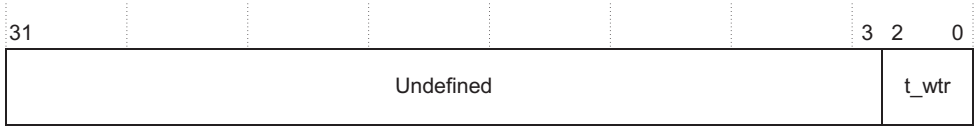


Figure 3-21 t_wtr Register bit assignments

Table 3-18 lists the 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 the write to read delay in memory clock cycles

3.3.17 t_xp Register

The read/write t_xp Register sets exit power-down command time in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-22 shows the register bit assignments. See also *Memory interface* on page 2-13.

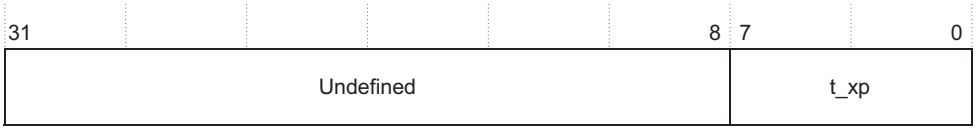


Figure 3-22 t_xp Register bit assignments

Table 3-19 lists the 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 the exit power-down command time in memory clock cycles

3.3.18 t_xsr Register

The read/write t_xsr Register sets exit self-refresh command time in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-23 shows the register bit assignments. See also *Memory interface* on page 2-13.

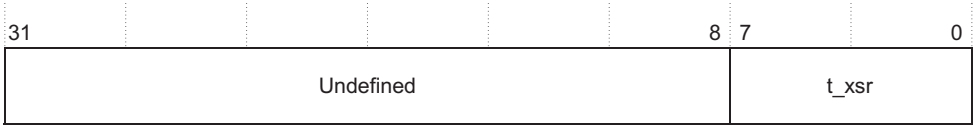


Figure 3-23 t_xsr Register bit assignments

Table 3-20 lists the 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 the exit self-refresh command time in memory clock cycles

3.3.19 t_esr Register

The read/write t_esr Register sets self-refresh command time in memory clock cycles. It can only be read from and written to in the Config or Low-power state. Figure 3-24 shows the register bit assignments. See also *Memory interface* on page 2-13.

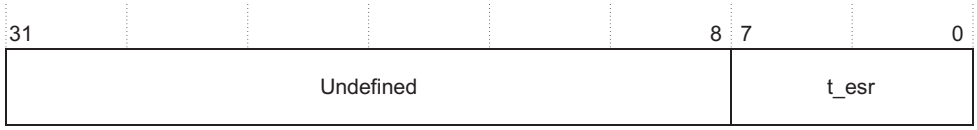


Figure 3-24 t_esr Register bit assignments

Table 3-21 lists the 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 command time in memory clock cycles

3.3.20 memory_cfg2 Register

The read/write memory_cfg2 Register determines the operating state of the memory controller and the memory. At reset the register value is set by the tie-off pins with the same names as the register names. After power-up you can change the register value through the APB interface. However if the memory controller is reset, the register value reverts to the tie-off port settings. See the *PrimeCell Dynamic Memory Controller (PL340) Integration Manual* for full information of the settings of these bits.

Figure 3-25 shows the register bit assignments.

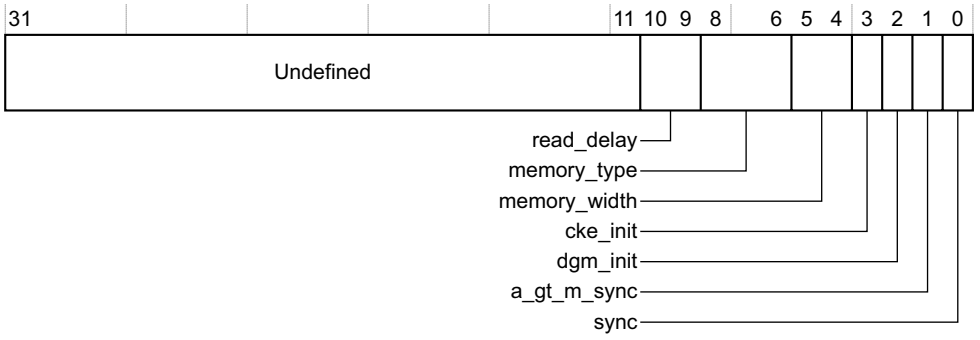


Figure 3-25 memory_cfg2 Register bit assignments

Table 3-22 lists the register bit assignments.

Table 3-22 memory_cfg2 Register bit assignments

Bits	Name	Function
[31:11]	-	Read undefined. Write as zero.
[10:9]	read_delay	Sets the latency in clock cycles of the pad interface.
[8:6]	memory_type	Sets the memory type, 0..3: 0 = SDR 1 = DDR 2 = eDRAM 3 = LPDDR. **Note** It is only legal to program the memory type between SDR and (LP)DDR for a memory controller configuration that supports it.
[5:4]	memory_width	Sets the width of the external memory: 2'b00 = 16-bit 2'b01 = 32-bit 2'b10 = 64-bit 2'b11 = Reserved. **Note** Only a memory width that is legal for the memory controller can be programmed.
[3]	cke_init	Sets the level for the **cke** outputs after reset.

Table 3-22 memory_cfg2 Register bit assignments (continued)

Bits	Name	Function
[2]	dqm_init	Sets the level for the **dqm** outputs after reset.
[1]	a_gt_m_sync	Requires to be set HIGH when running the **aclk** and **mclk** synchronously but with **aclk** running faster than **mclk**.
[0]	sync	Set high when **aclk** and **mclk** are synchronous.

3.3.21 memory_cfg3 Register

The read/write memory_cfg3 Register determines the operating state of the memory controller and the memory. At reset the register value is set by the tie-off pins with the same names as the register names. After power-up you can change the register value through the APB interface. However, if the memory controller is reset, the register value reverts to the tie-off port settings. See the *PrimeCell Dynamic Memory Controller (PL340) Integration Manual* for full information of the settings of these bits.

Figure 3-26 shows the register bit assignments.

Figure 3-26 memory_cfg3 Register bit assignments

Table 3-23 lists the register bit assignments.

Table 3-23 memory_cfg3 Register bit assignments

Bits	Name	Function
[31:12]	-	Read undefined. Write as zero.
[11:3]	prescale	Prescalar counter value.
[2:0]	max_outs_refs	Maximum number of outstanding refresh commands.

3.3.22 id_<n>_cfg Registers

The read/write id_<n>_cfg Registers are 16 registers that set the QoS and span address locations `0x100-0x200`. The registers can only be read from and written to in the Config or Low-power states.

Figure 3-27 shows the register bit assignments.

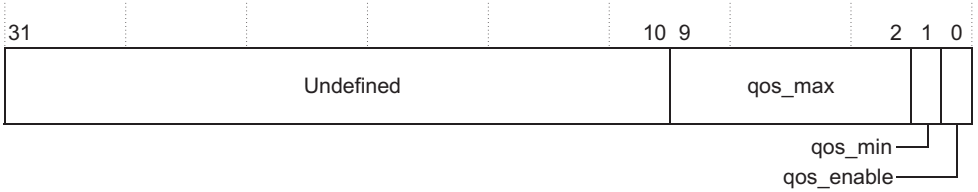


Figure 3-27 id_<n>_cfg Registers bit assignments

Table 3-24 lists the register bit assignments.

Table 3-24 id_<n>_cfg Registers bit assignments

Bits	Name	Function
[31:10]	-	Read undefined. Write as zero.
[9:2]	qos_max	Sets a maximum QoS.
[1]	qos_min	Sets a minimum QoS.
[0]	qos_enable	Enables a QoS value to be applied to memory reads from address ID n.

3.3.23 chip_<n>_cfg Registers

The read/write chip_<n>_cfg Registers are registers that set up the external memory device configuration. The number of external chips supported, and therefore the number of these registers, depends on your configuration. They span address locations `0x200-0x300`. There is one register per memory device. The registers configure the base address and address decoding method. The registers can only be read from and written to in the Config or Low-power states.

Figure 3-28 shows the register bit assignments.

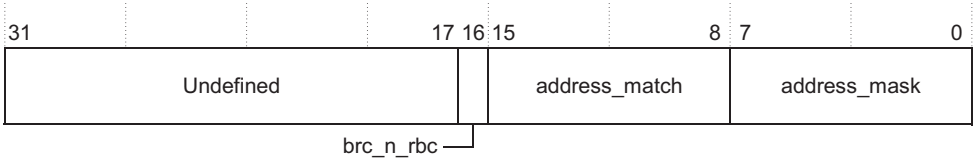


Figure 3-28 chip_<n>_cfg Registers bit assignments

Table 3-25 lists the register bit assignments.

Table 3-25 chip_<n>_cfg Registers bit assignments

Bits	Name	Function
[31:17]	-	Read undefined. Write as zero.
[16]	brc_n_rbc	Selects the memory organization as decoded from the AXI address: 1'b0 = Row, bank, column organization 1'b1 = Bank, row, column organization.
[15:8]	address_match	Comparison value for AXI address bits [31:24] to determine the chip that is selected.
[7:0]	address_mask	The mask for AXI address bits [31:24] to determine the chip that is selected: 1 = corresponding address bit is to be used for comparison.

3.3.24 user_status Register

This register returns the state of the **user_status[7:0]** primary inputs.

Table 3-26 lists the register bit assignments.

Table 3-26 user_status Registers bit assignments

Bits	Name	Function
[7:0]	user_status	Value of primary input pins

3.3.25 user_config Register

This register sets the value of the **user_config[7:0]** primary outputs.

Table 3-27 lists the register bit assignments.

Table 3-27 user_config Registers bit assignments

Bits	Name	Function
[7:0]	user_config	Sets primary output pins

3.3.26 Peripheral Identification Registers 0-3

The periph_id Registers are four 8-bit read-only registers, that span address locations 0xFE0-0xFEC. The registers can conceptually be treated as a single register that holds a 32-bit peripheral ID value. An external master reads them to determine the memory controller version of the device.

Table 3-28 lists the register bit assignments.

Table 3-28 periph_id Register bit assignments

Bits	Name	Description
[31:24]	integration_cfg	Configuration options are peripheral-specific. See *Peripheral Identification Register 3* on page 3-30.
[23:20]	-	The peripheral revision number is revision-dependent.
[19:12]	designer	Designer's ID number. This is 0x41 for ARM.
[11:0]	part_number	Identifies the peripheral. The part number for PL340 is 0x340.

Figure 3-29 on page 3-28 shows the correspondence between bits of the periph_id registers and the conceptual 32-bit Peripheral ID Register.

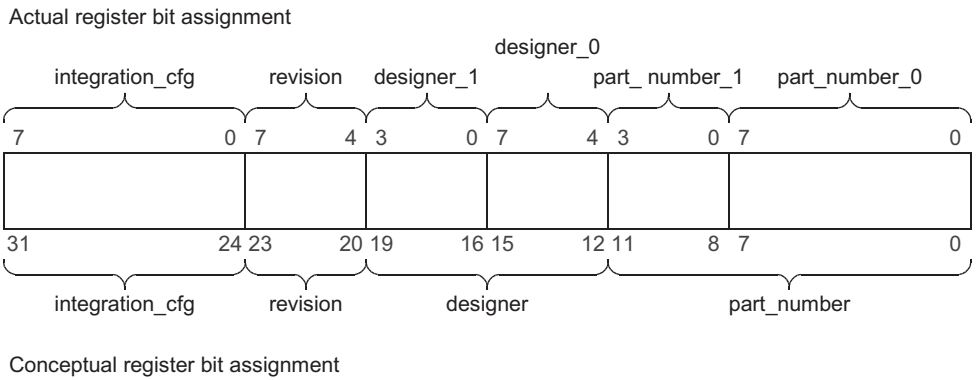


Figure 3-29 periph_id Register bit assignments

The periph_id Registers are described in:

- *Peripheral Identification Register 0*
- *Peripheral Identification Register 1* on page 3-29
- *Peripheral Identification Register 2* on page 3-30

- *Peripheral Identification Register 3* on page 3-30.

Peripheral Identification Register 0

The periph_id_0 Register is hard-coded and the fields within the register determine the reset value. Table 3-29 lists the register bit assignments.

Table 3-29 periph_id_0 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined
[7:0]	part_number_0	These bits read back as `0x40`

Peripheral Identification Register 1

The periph_id_1 Register is hard-coded and the fields within the register determine the reset value. Table 3-30 lists the register bit assignments.

Table 3-30 periph_id_1 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined
[7:4]	designer_0	These bits read back as `0x1`
[3:0]	part_number_1	These bits read back as `0x3`

Peripheral Identification Register 2

The periph_id_2 Register is hard-coded and the fields within the register determine the reset value. Table 3-31 lists the register bit assignments.

Table 3-31 periph_id_2 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined.
[7:4]	revision	These bits read back as the revision number. This can be between 0 and 15: • `0x0` is r0p0 • `0x1` is r1p0. • `0x2` is r2p0.
[3:0]	designer_1	These bits read back as `0x4`.

Peripheral Identification Register 3

The periph_id_3 register is hard-coded and the fields within the register determine the reset value. Table 3-32 lists the register bit assignments.

Table 3-32 periph_id_3 Register bit assignments

Bits	Name	Description
[31:8]	-	Read undefined.
[7:4]	-	Reserved for future use. Read undefined.
[3:0]	Customer Modified	Customer-modified number. 0 from ARM.

3.3.27 PrimeCell Identification Registers 0-3

The pcell_id Registers are four 8-bit wide read-only registers, that span address locations `0xFF0-0xFFC`. The registers can be treated conceptually as a single register that holds a 32-bit PrimeCell identification value. You can use the register for automatic BIOS configuration. The pcell_id Register is set to `0xB105F00D`.

You can access the register with one wait state.

Figure 3-30 shows the register bit assignments.

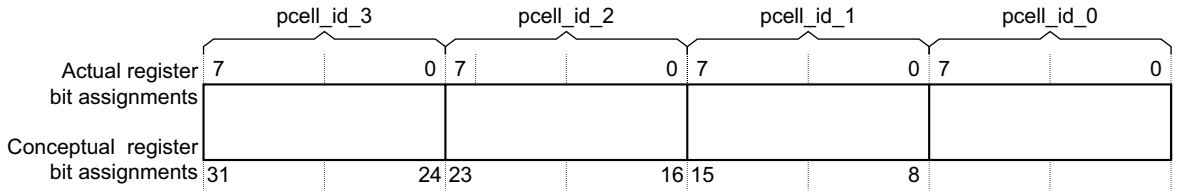


Figure 3-30 pcell_id Register bit assignments

Table 3-33 lists the register bit assignments.

Table 3-33 pcell_id Register bit assignments

Component ID register		CompID0-3 register		
Bits	**Reset value**	**Register**	**Bits**	**Description**
-	-	pcell_id_3	[31:8]	Read undefined
[31:24]	`0xB1`	pcell_id_3	[7:0]	These bits read back as `0xB1`
-	-	pcell_id_2	[31:8]	Read undefined
[23:16]	`0x05`	pcell_id_2	[7:0]	These bits read back as `0x05`
-	-	pcell_id_1	[31:8]	Read undefined
[15:8]	`0xF0`	pcell_id_1	[7:0]	These bits read back as `0xF0`
-	-	pcell_id_0	[31:8]	Read undefined
[7:0]	`0x0D`	pcell_id_0	[7:0]	These bits read back as `0x0D`

Chapter 4
Programmer's Model for Test

This chapter describes the additional logic for functional verification and production testing. It contains the following section:

- *Integration test registers* on page 4-2.

4.1 Integration test registers

Test registers are provided for integration testing.

Figure 4-1 shows the Integration Test Register map.

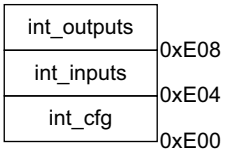


Figure 4-1 Integration Test Register map

Table 4-1 lists the integration test registers.

Table 4-1 Memory controller test register summary

Name	Base offset	Type	Reset value	Description
int_cfg	0xE00	RW	0x0	*Integration Configuration Register*
int_inputs	0xE04	RO	-[a]	*Integration Inputs Register* on page 4-3
int_outputs	0xE08	WO	-	*Integration Outputs Register* on page 4-4

a. Dependent on tie-off ports.

4.1.1 Integration Configuration Register

The read/write int_cfg Register selects the integration test registers. This register is only for test. It can only be read from and written to in Config state.

Figure 4-2 shows the register bit assignments.

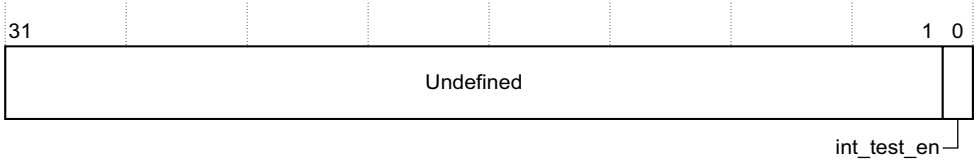


Figure 4-2 int_cfg Register bit assignments

Table 4-2 lists the register bit assignments.

Table 4-2 int_cfg Register bit assignments

Bits	Name	Function
[31:1]	Undefined	Read undefined. Write as zero.
[0]	int_test_en	When set, outputs are driven from the integration test registers, and input integration register values show external port states.

4.1.2 Integration Inputs Register

The read-only int_inputs Register enables an external master to access the inputs of the memory controller using the APB interface. This register is only for test. It can only be read in Config state.

Figure 4-3 shows the register bit assignments.

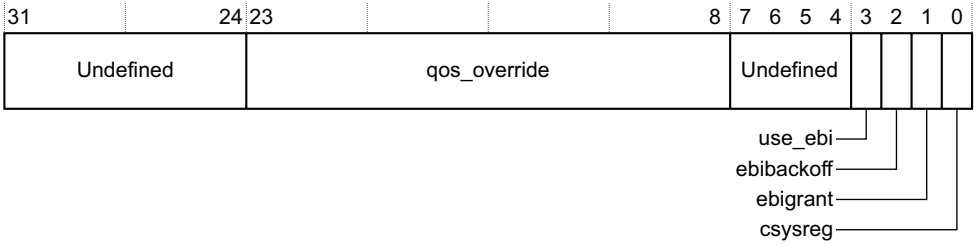


Figure 4-3 int_inputs Register bit assignments

Table 4-3 lists the register bit assignments.

Table 4-3 int_inputs Register bit assignments

Bits	Name	Function
[31:24]	-	Read undefined
[23:8]	qos_override	Returns the value of the **qos_override** external input
[7:4]	-	Read undefined
[3]	use_ebi	Returns the value of the **use_ebi** external input

Table 4-3 int_inputs Register bit assignments (continued)

Bits	Name	Function
[2]	ebibackoff	Returns the value of the **ebibackoff** external input
[1]	ebigrant	Returns the value of the **ebigrant** external input
[0]	csysreq	Returns the value of the **csysreq** external input

4.1.3 Integration Outputs Register

The write-only int_outputs Register enables an external master to access the outputs of the memory controller using the APB interface. It can only be read in Config state.

Figure 4-4 shows the register bit assignments.

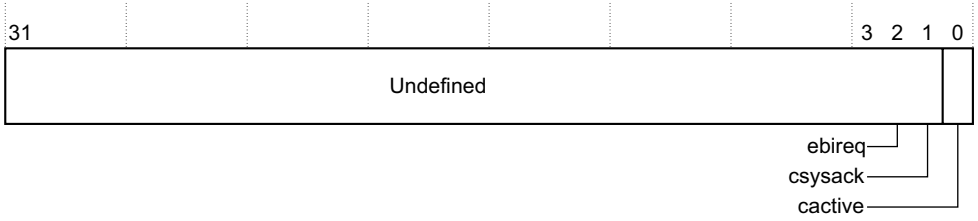


Figure 4-4 int_outputs Register bit assignments

Table 4-4 lists the register bit assignments.

Table 4-4 int_outputs Register bit assignments

Bits	Name	Function
[31:3]	-	Undefined. Write as zero.
[2]	ebireq	Drives the **ebireq** external output.
[1]	csysack	Drives the **csysack** external output.
[0]	cactive	Drives the **cactive** external output.

Chapter 5
Device Driver

This chapter describes the device driver. It contains the following section:

- *Device driver functions* on page 5-2.

5.1 Device driver functions

The memory controller driver contains the set of functions that enable:

- initialization of the memory controller
- initialization of the memory devices
- control of the memory controller state
- reading of the status of the memory controller.

5.1.1 Register access functions

The header `appl340.h` defines these functions and `pl340.c` contains the source code.

apPL340_StatusGet

This function reads the status bits of the Memory Controller Status Register to determine the current state of the memory controller.

apPL340_FixedConfigGet

This function reads the hardware configuration from the Memory Controller Status Register.

apPL340_CommandSet

This function writes commands to the Memory Controller Command Register that change the state of the memory controller.

apPL340_DirectCommandSet

This function writes data to the Direct Command Register that generates commands to external memory.

apPL340_MemConfigSet

This function writes memory configuration data to the Memory Controller Configuration Register that defines the structure and features of the memory.

apPL340_MemConfigGet

This function is the complement of `apPL340_MemConfigSet`, to read the configuration data from the Memory Controller Configuration Register and format it in the same way as that passed to `apPL340_MemConfigSet`.

apPL340_MemConfig2Set

This function writes memory configuration data to the Memory Controller Configuration Register that defines the structure and features of the memory.

apPL340_MemConfig2Get

This function is the complement of `apPL340_MemConfigSet`, to read the configuration data from the Memory Controller Configuration Register and format it in the same way as that passed to `apPL340_MemConfigSet`.

apPL340_MemConfig3Set

This function writes memory configuration data to the Memory Controller Configuration Register that defines the structure and features of the memory.

apPL340_MemConfig3Get

This function is the complement of `apPL340_MemConfigSet`, to read the configuration data from the Memory Controller Configuration Register and format it in the same way as that passed to `apPL340_MemConfigSet`.

apPL340_RefreshPeriodSet

This function sets the refresh period to the Refresh Period Register.

apPL340_RefreshPeriodGet

This function is the complement of `apPL340_RefreshPeriodSet`, to read the refresh period from the memory controller Refresh Period Register.

apPL340_TimingConfigSet

This function writes timing configuration data to the Memory Controller Timing Registers.

apPL340_TimingConfigGet

This function is the complement of ap`PL340_TimingConfigSet`, to read timing data from Memory Controller Timing Registers and format it in the same way as that passed to `apPL340_TimingConfigSet`.

apPL340_IdConfigSet

This function writes QoS data to the id_<n>_cfg Register.

apPL340_IdConfigGet

This function is the complement of `apPL340_IdConfigSet`, to read QoS data from the id_<n>_cfg Register.

apPL340_ChipConfigSet

This function writes the external memory device configuration to the chip_<n>_cfg Registers of the memory controller.

apPL340_ChipConfigGet

This function is the complement of `apPL340_ChipConfigSet`, to read the configuration of external memory from the chip_<n>_cfg Registers in the memory controller.

apPL340_UserStatusGet

This function reads the user-defined user_status Register.

apPL340_UserConfigSet

This function writes to the user-defined user_config Register.

apPL340_Initialize

This function is part of the PrimeCell driver framework and initializes the memory controller driver.

apPL340_StateSizeGet

This function is part of the PrimeCell driver framework and returns the memory required for the state structure in the PrimeCell driver.

5.1.2 Example memory initialization functions

These functions show typical initialization sequences for the memory, and are contained in `appl340tst.c`. These functions implement the example setup that *Example MDDR setup* on page 2-27 shows.

apPL340_Initialization

This function initializes the memory controller with memory-specific data that is passed to the register access functions. This sequence of function calls sets up:

1. The timing parameters.
2. Memory configuration.
3. Chip selects for the memory system.

apPL340_InitializeMemory

This function initializes the memory mode registers by calling the `apPL340_DirectCommandSet` function multiple times, and performing a memory-specific sequence of commands to set up the memory.

This function must only be called after `apPL340_Initialization` because that function sets up the mapping of the Direct Command Register to the physical addressing of external memory.

Appendix A
Signal Descriptions

This chapter describes the AMBA AXI, AMBA APB, and module-specific non-AMBA signals used with the memory controller. It contains the following sections:

A.1 Clock and reset signals

Table A-1 lists the clock and reset signals.

Table A-1 Clock and reset signals

Signal	Type	Source/ destination	Description
aclk	Input	Clock source	Clock for the **aclk** domain.
aresetn	Input	Reset source	**aclk** domain reset signal. This signal is active LOW.
mclk	Input	Clock source	Clock for **mclk** domain.
mclkn	Input	Clock source	Optional clock for pad interface block.
mclkx2	Input	Clock source	Optional clock for pad interface block.
mclkx2n	Input	Clock source	Optional clock for pad interface block.
mresetn	Input	Reset source	Reset for **mclk** domain. This signal is active LOW.

A.2 Miscellaneous signals

Table A-2 lists the miscellaneous signals.

Table A-2 Miscellaneous signals

Signal	Type	Source	Description
a_gt_m_sync	Input	Tie-off pin	When HIGH, indicates **aclk** is greater than **mclk** but is still synchronous.
sync	Input	Tie-off pin	When HIGH, indicates **aclk** is synchronous to **mclk**. Otherwise, they are asynchronous.
cke_init	Input	Tie-off pin	The **cke** output port to the external memory resets to this value.
dft_en_clk_out	Input	Tie-off pin	This signal is used for ATPG testing only.
dqm_init	Input	Tie-off pin	The **dqm** output ports to the external memory reset to this value.
memory_width[1:0]	Input	Tie-off pin	These configure the external memory width. See Table 3-2 on page 3-7 for the definition of these two pins. **Note** You can use each PL340 configuration at half of its memory width, by setting the memory_width tie-off, provided that: • the new memory width is not less than 16 bits • the effective memory width is not less than half the AXI interface width.
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.
rst_bypass	Input	Tie-off pin	This signal is used for ATPG testing only.
use_ebi	Input	Tie-off pin	When HIGH, indicates that the memory controller must operate with a PrimeCell EBI (PL220).
user_config[7:0]	Output	External control logic	General purpose APB-accessible control pins. For example, you can use this for DLL control.

Table A-2 Miscellaneous signals (continued)

Signal	Type	Source	Description
user_status[7:0]	Input	External control logic	General-purpose APB-accessible input pin.
memory_type[2:0]	Input	Tie-off pin	Selects the memory type of the current configuration. Not all values are valid for a single configuration. You can override this value using the APB interface.
read_delay[1:0]	Input	Tie-off pin	This value varies the delay permitted before the capture of read data into the memory clock domain from the memory device.

A.3 AXI signals

The following sections describe the AXI signals:

- *Write address channel signals*
- *Write data channel signals* on page A-6
- *Buffered 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-10.

A.3.1 Write address channel signals

Table A-3 lists the AXI write address channel signals.

Table A-3 Write address channel signals

Signal	Type	Source/ destination	Description
awid[7:0]	Input	Master	Address ID. This signal is the ID tag of the read or write transaction.
awaddr[31:0]	Input	Master	Address. The address bus gives the address of the first transfer in a burst. The associated control signals determine the addresses of the remaining transfers in the burst.
awlen[3:0]	Input	Master	Burst length. This signal indicates the number of transfers in a burst.
awsize[2:0]	Input	Master	Burst size. This signal indicates the size of each transfer in a burst. For write transfers, byte lane strobes indicate the byte lanes to update in memory.
awburst[1:0]	Input	Master	Burst type. This signal indicates a fixed, incrementing, or wrapping burst. The AXI uses **aburst[1:0]** and **asize[2:0]** to calculate the address for successive transfers in the burst.
awlock[1:0]	Input	Master	Lock type. This signal indicates a normal, exclusive, or locked transaction.
awcache[3:0]	Input	Master	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.

Table A-3 Write address channel signals (continued)

Signal	Type	Source/ destination	Description
awprot[1:0]	Input	Master	Protection level. This signal indicates the normal, privileged, or secure protection level of the transaction, and whether the transaction is a data access or an instruction access.
awvalid	Input	Master	Address valid. This signal indicates that valid address and control information are available: 0 = address and control information not available 1 = address and control information available. The address and control information remain stable until the address acknowledge signal, **aready**, goes HIGH.
awready	Output	Slave	Address ready. This signal indicates that the slave is ready to accept an address: 0 = slave not ready 1 = slave ready.

A.3.2 Write data channel signals

Table A-4 lists the AXI write data channel signals.

Table A-4 Write data channel signals

Signal	Type	Source/ destination	Description
wid[7:0]	Input	Master	Write ID tag. This signal is the ID tag of the write transfer. The **wid** value must match the **awid** value of the write transaction.
wdata[PORTWIDTH-1:0][a]	Input	Master	Write data. The write data bus can be 32, 64 or 128 bits wide.
wstrb[PORTBYTES-1:0][b]	Input	Master	Write strobes. This signal indicates the byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus. Therefore, **wstrb[n]** corresponds to **wdata[(8 ° n) + 7:(8 ° n)]**.

Table A-4 Write data channel signals (continued)

Signal	Type	Source/ destination	Description
wlast	Input	Master	Write last. This signal indicates the last transfer in a write burst.
wvalid	Input	Master	Write valid. This signal indicates that valid write data and strobes are available: 0 = write data and strobes not available 1 = write data and strobes available.
wready	Output	Slave	Write ready. This signal indicates that the slave can accept the write data: 0 = slave not ready 1 = slave ready.

a. PORTWIDTH is the data width of the AXI port in bits.
b. PORTBYTES is the data width of the AXI port in bytes.

A.3.3 Buffered write response channel signals

Table A-5 lists the buffered write response channel signals.

Table A-5 Buffered write response channel signals

Signal	Type	Source/ destination	Description
bid[7:0]	Output	Slave	Response ID. The identification tag of the write response. The **bid** value must match the **awid** value of the write transaction to which the slave is responding.
bresp[1:0]	Output	Slave	Write response. This signal indicates the status of the write transaction. The permitted responses are OKAY and EXOKAY.
bvalid	Output	Slave	Write response valid. This signal indicates that a valid write response is available: 0 = write response not available 1 = write response available.
bready	Input	Master	Response ready. This signal indicates that the master can accept the response information: 0 = master not ready 1 = master ready.

A.3.4 Read address channel signals

Table A-6 lists the AXI read address channel signals.

Table A-6 Read address channel signals

Signal	Type	Source/ destination	Description
arid[7:0]	Input	Master	Address ID. This signal is the ID tag of the read or write transaction.
araddr[31:0]	Input	Master	Address. The address bus gives the address of the first transfer in a burst. The associated control signals determine the addresses of the remaining transfers in the burst.
arlen[3:0]	Input	Master	Burst length. This signal indicates the number of transfers in a burst.
arsize[2:0]	Input	Master	Burst size. This signal indicates the size of each transfer in a burst. For write transfers, byte lane strobes indicate the byte lanes to update in memory.
arburst[1:0]	Input	Master	Burst type. This signal indicates a fixed, incrementing, or wrapping burst. The AXI uses **aburst[1:0]** and **asize[2:0]** to calculate the address for successive transfers in the burst.
arlock[1:0]	Input	Master	Lock type. This signal indicates a normal, exclusive, or locked transaction.
arcache[3:0]	Input	Master	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.
arprot[1:0]	Input	Master	Protection level. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access, or an instruction access.
arvalid	Input	Master	Address valid. This signal indicates that valid address and control information are available: 0 = address and control information not available 1 = address and control information available. The address and control information remain stable until the address acknowledge signal, **aready**, goes HIGH.
arready	Output	Slave	Address ready. This signal indicates that the slave is ready to accept an address: 0 = slave not ready 1 = slave ready.

A.3.5 Read data channel signals

Table A-7 lists the AXI read data channel signals.

Table A-7 Read data channel signals

Signal	Type	Source/ destination	Description
rid[7:0]	Output	Slave	Read ID tag. This signal is the ID tag of the read transfer. The **rid** value must match the **arid** value of the read transaction to which the slave is responding.
rdata[PORTWIDTH-1:0][a]	Output	Slave	Read data. The read data bus can be 32, 64, or 128 bits wide.
rresp[1:0]	Output	Slave	Read response. This signal indicates the status of the read transfer. The permitted responses are `OKAY` and `EXOKAY`.
rlast	Output	Slave	Read last. This signal indicates the last transfer in a read burst.
rready	Input	Master	Read ready. This signal indicates that the master can accept the read data and response information: 0 = master not ready 1= master ready. 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 device has a sufficiently high system priority to prevent this.
rvalid	Output	Slave	Read valid. This signal indicates that valid read data is available: 0 = read data not available 1 = read data available.

a. PORTWIDTH is the data width of the AXI port in bits.

A.3.6 AXI low-power interface signals

Table A-8 lists the optional low-power interface signals.

Table A-8 AXI low-power interface signals

Signal	Type	Source/ destination	Description
cclken	Input	Bus clock	Clock enable.
csysreq	Input	Clock controller	System low-power request. This signal is a request from the system clock controller for the peripheral to enter a Low-power state.
csysack	Output	Peripheral device	Low-power request acknowledgement. This signal is the acknowledgement from a peripheral of a system Low-power request.
cactive	Output	Peripheral device	Clock active. This signal indicates that the peripheral requires its clock signal: 0 = peripheral clock not required 1 = peripheral clock required.

A.4 APB signals

Table A-9 lists the APB signals.

Table A-9 APB signals

Signal	Type	Source/ destination	Description
paddr[11:0]	Input	APB address bus	This is the APB address bus, and can be up to 32-bits wide, and is driven by the peripheral bus bridge unit.
pclken	Input	Bus clock	Clock enable for **clk** in APB domain.
penable	Input	APB strobe	This strobe signal times all accesses on the peripheral bus. The enable signal indicates the second cycle of an APB transfer. The rising edge of **penable** occurs in the middle of the APB transfer.
prdata[31:0]	Output	APB read data bus	The read data bus is driven by the selected slave during read cycles, when **pwrite** is LOW. The read data bus can be up to 32-bits wide.
pready	Output	APB	APB transfer wait signal.
psel	Input	APB select	A signal from the secondary decoder, within the peripheral bus bridge unit, to each peripheral bus slave x. This signal indicates that the slave device is selected, and a data transfer is required. There is a **psel** signal for each bus slave.
pslverr	Output	APB transfer error	This is tied-off within the memory controller. Included for completeness.
pwdata[31:0]	Input	APB write data bus	The write data bus is driven by the peripheral bus bridge unit during write cycles, when **pwrite** is HIGH. The write data bus can be up to 32-bits wide.
pwrite	Input	APB transfer direction	When HIGH, this signal indicates an APB write access and when LOW, a read access.

A.5 Pad interface signals

Table A-10 lists the APB signal names.

Table A-10 Pad interface signals

Signal	Type	Source/ destination	Description
add[15:0]	Output	External memory	Address bus
ap	Output	External memory	Auto precharge bit
ba[1:0]	Output	External memory	Bank select
cas_n	Output	External memory	Column address strobe
cke[MEMORIES-1:0][a]	Output	External memory	Clock enable
clk_out[MEMORIES-1:0][b]	Output	External memory	Memory clock
cs_n[MEMORIES-1:0]	Output	External memory	Chip select
data_en	Output	External memory	Data direction enable
dqm[MEMBYTES-1:0][c]	Output	External memory	Data bus mask bits
dqs_in_<n>	Input	External memory	Data strobe in, DDR only
dq_in[MEMWIDTH-1:0][d]	Input	External memory	Data bus input
dqs_in_n_<n>	Input	External memory	Data strober bar, DDR only
dqs_out_<n>	Output	External memory	Data strobe out, DDR only
dq_out[MEMWIDTH-1:0]	Output	External memory	Data bus output
fbclk_in	Input	External memory	Feedback clock
ras_n	Output	External memory	Row address strobe
we_n	Output	External memory	Write enable

a. **cke** can be configured for one option, that is, global **cke** for all memory devices or a local **cke** pin per memory device.
b. MEMORIES is the number of chip selects.
c. MEMBYTES is the data width of the external memory bus in bytes.
d. MEMWIDTH is the data width of the external memory bus in bits.

A.6 Memory Controller EBI signals

Table A-11 lists the memory controller EBI signals.

Table A-11 EBI signals

Signal	Type	Source/ destination	Description
ebibackoff	Input	DMC	External memory bus access backoff. The EBI backoff signal goes active HIGH when the EBI wants to remove the memory controller from the memory bus so that another memory controller can be granted the memory bus.
ebigrant	Input	DMC	External memory bus grant. The EBI grant signal goes HIGH when the EBI grants the external memory bus.
ebireq	Output	External	External memory bus request. The EBI request signal goes HIGH when the memory controller requires the memory bus.

Glossary

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

Abort

A mechanism that indicates to a core that the value associated with a memory access is invalid. An abort can be caused by the external or internal memory system as a result of attempting to access invalid instruction or data memory. An abort is classified as either a Prefetch or Data Abort, and an internal or External Abort.

See also Data Abort, External Abort and Prefetch Abort.

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 High-performance Bus (AHB)
A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM Limited recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

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 details 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.

AHB *See* Advanced High-performance Bus.

Aligned A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

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.

AXI *See* Advanced eXtensible Interface.

Big-endian Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory.

See also Little-endian and Endianness.

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.

Doubleword A 64-bit data item. The contents are taken as being an unsigned integer unless otherwise stated.

Endianness Byte ordering. The scheme that determines the order that successive bytes of a data word are stored in memory. An aspect of the system's memory mapping.

See also Little-endian and Big-endian.

Little-endian Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory.

See also Big-endian and Endianness.

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.

Unaligned A data item stored at an address that is not divisible by the number of bytes that defines the data size is said to be unaligned. For example, a word stored at an address that is not divisible by four.

Undefined Indicates an instruction that generates an Undefined instruction trap. See the *ARM Architecture Reference Manual* for more details on ARM exceptions.

Unpredictable Means that the behavior of the ETM cannot be relied on. Such conditions have not been validated. When applied to the programming of an event resource, only the output of that event resource is Unpredictable. Unpredictable behavior can affect the behavior of the entire system, because the ETM is capable of causing the core to enter debug state, and external outputs can be used for other purposes.

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.