CoreLink™ SMC-35x AXI Static Memory Controller Series

Revision: r2p2

Technical Reference Manual

ARM®

ARM DDI 0380H (ID122111)

CoreLink SMC-35x AXI Static Memory Controller Series
Technical Reference Manual

Release Information

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

Change history

Date	Issue	Confidentiality	Change
26 August 2005	A	Non-Confidential	First release
08 March 2006	B	Non-Confidential	Updated for r1p0, configurable IP
15 June 2006	C	Confidential	First release for r1p1
22 December 2006	D	Non-Confidential	First release for r1p2
25 May 2007	E	Non-Confidential	First release for r2p0
20 July 2007	F	Non-Confidential	Maintenance update for r2p0
12 October 2007	G	Non-Confidential	First release for r2p1
11 November 2011	H	Non-Confidential	First release for r2p2

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
CoreLink SMC-35x AXI Static Memory Controller Series Technical Reference Manual

Preface

This preface introduces the *CoreLink SMC-35x AXI Static Memory Controller Series Technical Reference Manual* (TRM). It contains the following sections:

About this book

This is the TRM for the CoreLink SMC-35x AXI Static Memory Controller Series. The *Static Memory Controller* (SMC) product range comprises a number of configurable memory controllers that support SRAM and NAND on the memory interface.

Product revision status

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

r*n* Identifies the major revision of the product.

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

Intended audience

This book is written for implementation engineers and architects. It provides a description of an optimal SMC architecture. The SMC product range provides an interface between the *Advanced eXtensible Interface* (AXI) system bus and off-chip memory devices.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the SMC product range and its features.

Chapter 2 *Functional Description*

Read this for an overview of the major functional blocks and the operation of the SMC product range.

Chapter 3 *Programmers Model*

Read this for a description of the registers.

Chapter 4 *Programmers Model for Test*

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

Chapter 5 *Device Driver Requirements*

Read this for a description of device driver requirements.

Chapter 6 *Configurations*

Read this for a description of non-universal SMC configurations.

Appendix B *Signal Descriptions*

Read this for a description of the signals.

Appendix A *Revisions*

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

Glossary

The *ARM Glossary* is a list of terms used in ARM documentation, together with definitions for those terms. The *ARM Glossary* does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

See *ARM Glossary*, `http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html`.

Typographical conventions

Conventions that this book uses are described in:

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

Typographical

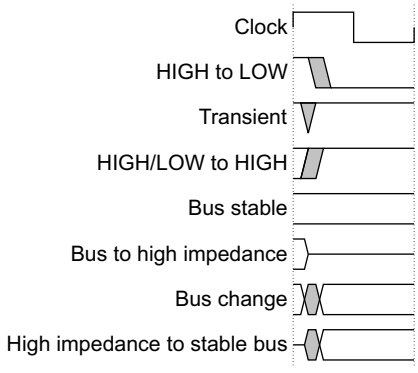
The typographical conventions are:

italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace`	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

Timing diagrams

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

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



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
- HIGH for active-HIGH signals
- LOW for active-LOW signals.

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

Additional reading

This section lists publications by ARM and by third parties.

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

ARM publications

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

- *CoreLink SMC-35x AXI Static Memory Controller Series Integration Manual* (ARM DII 0137)
- *CoreLink SMC-35x AXI Static Memory Controller Series Implementation Guide* (ARM DII 0138)
- *CoreLink SMC-35x AXI Static Memory Controller Series Supplement to AMBA Designer (FD001) User Guide* (ARM DSU 0006)
- *AMBA® Designer (FD001) User Guide* (ARM DUI 0333)
- *AMBA AXI Protocol Specification* (ARM IHI 0022)
- *AMBA 3 APB Protocol Specification* (ARM IHI 0024)
- *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual* (ARM DDI 0249).

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

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

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

Feedback on content

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

- the title
- the number, ARM DDI 0380H
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

This chapter introduces the SMC-35x series. It contains the following sections:

1.1 About the SMC-35x series

The SMC is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral. The product range consists of high-performance, area-optimized SRAM and NAND memory controllers with on-chip bus interfaces that conform to the AMBA *Advanced eXtensible Interface* (AXI) protocol.

The product range consists of a number of controllers that support either one or two memory interfaces of type NAND or SRAM. The controller variants are:

SMC-351 Single NAND interface.

SMC-352 Single SRAM interface.

SMC-353 SRAM and NAND interfaces.

SMC-354 Dual SRAM interfaces.

The NAND memory interface type is defined as supporting NAND flash with multiplexed *Address/Data* (A/D) buses.

The SRAM memory interface type is defined as supporting:

- synchronous or asynchronous SRAM
- *Pseudo Static Random Access Memory* (PSRAM)
- NOR flash
- NAND flash devices with an SRAM interface.

You can configure aspects of the SMC-35x series to provide the optimum features, performance, and gate count required for your intended application. For a summary of the configurable features supported, see *Features of the SMC-35x series* on page 1-3.

Figure 1-1 shows the interfaces of the SMC-35x series product range.

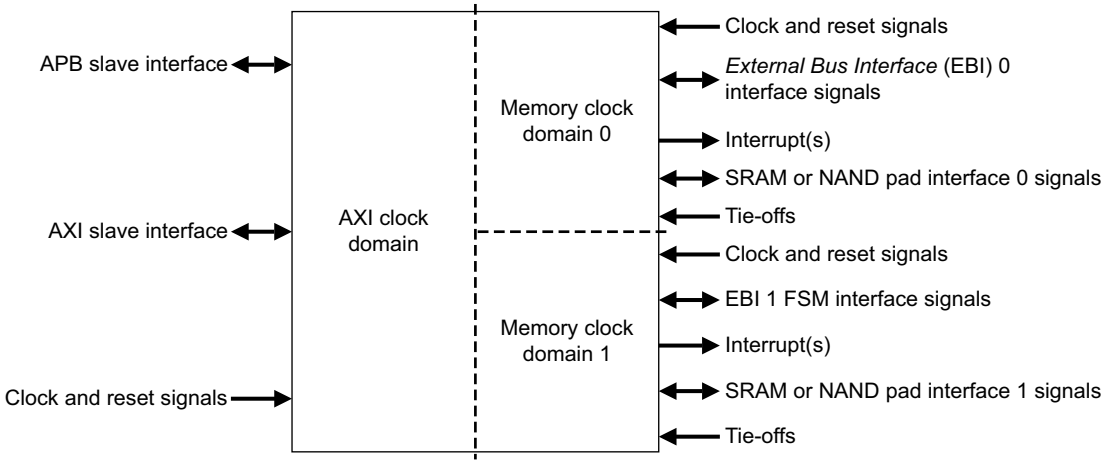


Figure 1-1 SMC interfaces

Figure 1-2 on page 1-3 shows an example system containing the SMC-351 variant.

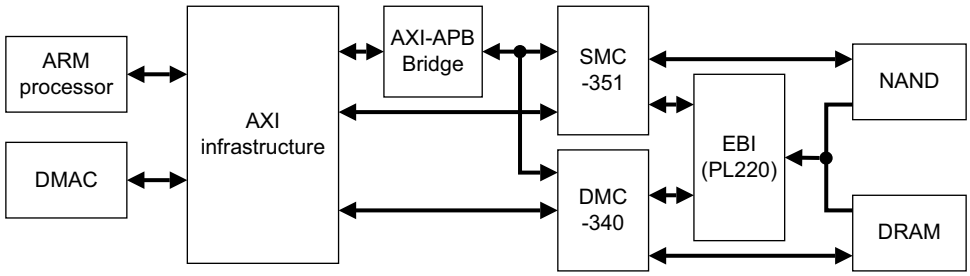


Figure 1-2 SMC-351 example system

The system contains:

- two bus masters:
  - an ARM processor
  - a *DMA Controller* (DMAC).
- AXI infrastructure component
- two memory controllers:
  - SMC-351
  - a *Dynamic Memory Controller* (DMC).
- an *External Bus Interface* (EBI).

The AXI interconnect enables each bus master to access both bus slaves. To reduce pin count, the EBI multiplexes the address and data pins of the SMC and DMC memory interfaces.

Figure 1-3 shows an example system containing the SMC-353 variant.

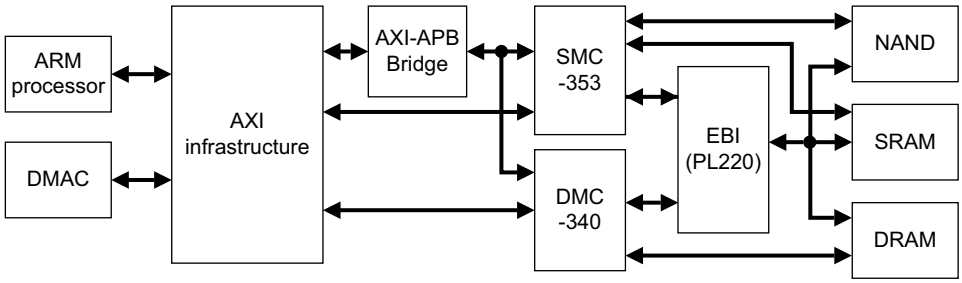


Figure 1-3 SMC-353 example system

This configuration of the SMC has two memory interfaces:

- one is connected to NAND flash
- one is connected to SRAM memory.

The EBI enables the address and data pins of three memory interfaces to be multiplexed. In this case, the three memory interfaces are:

- the NAND interface of the SMC-353
- the SRAM interface of the SMC-353
- the dynamic memory interface of the DMC-340.

1.1.1 Features of the SMC-35x series

The SMC provides the following features:

- You can configure the SMC to support the following options:
  - Maximum NAND memory data widths of 8 bits or 16 bits.

- Maximum SRAM memory data widths of 8 bits, 16 bits, or 32 bits.
- AXI data width of 32 bits or 64 bits.
- Up to four chip selects per memory interface.
- Configurable command, read data, and write data FIFO depths.
- An additional pipeline stage within the format logic that enables higher AXI clock frequencies at the cost of an additional clock cycle of latency.
- Configurable number of outstanding exclusive accesses supported.
- Optional 2-bit detection, 1-bit correction *Error Correction Code* (ECC) block for *single-level cell* (SLC) NAND memories. This identifies the occurrence and location of errors, enabling software to correct them.

Table 1-1 and Table 1-2 show the supported memory widths and AXI data widths. You can program the memory width for each chip select.

Table 1-1 SRAM memory widths and AXI data widths

AXI width	Configured memory width	Supported memory width
32 bits	32 bits	32 bits, 16 bits
32 bits	16 bits	16 bits, 8 bits
32 bits	8 bits	8 bits
64 bits	32 bits	32 bits, 16 bits
64 bits	16 bits	16 bits
64 bits	8 bits	Not supported

Table 1-2 NAND memory widths and AXI data widths

AXI width	Configured memory width	Supported memory width
32 bits	32	Not supported
32 bits	16 bits	16 bits, 8 bits
32 bits	8 bits	8 bits
64 bits	32 bits	Not supported
64 bits	16 bits	16 bits, 8 bits
64 bits	8 bits	8 bits

- Support for *ARM Architecture Version 6* (ARMv6) exclusive access transfers to SRAM.
- Programmable interrupt generation to indicate NAND flash status.
- Programmable cycle timings, and memory width per chip select.
- Programmable address cycles and command values for NAND flash accesses enabling operation with a variety of NAND devices.
- Atomic switching of memory device and controller operating modes.
- Support for the *PrimeCell EBI (PL220)* that enables sharing of external address and data bus pins between memory controller interfaces.
- Support for AXI low-power interface.

- Support for a **remap** signal for each interface.
- Support for multiple clock domains and configurable to be synchronous or asynchronous.

1.1.2 AXI interface attributes

The slave interface has the following attributes, that are fixed for a particular configuration of the SMC:

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 indicates where the slave has combined read and write transaction storage so this is mutually exclusive to the write and 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 the slave can transmit data. This is counted from the earliest transaction.

Table 1-3 shows the attribute formats.

Table 1-3 Attribute formats

Attribute	Value
Combined acceptance capability	2 for SMC-351 and SMC-352 3 for SMC-353 and SMC-354
Write interleave depth	1
Read data reorder depth	Read acceptance capability

1.2 Supported devices

The release note supplied with the SMC deliverables lists the static memory devices that ARM tests with the SMC variant.

Some memory devices, or series of memory devices, have specific limitations:

Intel W18 series NOR FLASH, for example 28F128W18TD

These devices, when in synchronous operation, use a **WAIT** pin. However, non-array operations, when in synchronous mode, do not use the **WAIT** pin and it is always asserted. The SMC cannot differentiate between array and non-array accesses and therefore cannot support these non-array accesses.

Therefore, W18 devices can only carry out non-array operations such as `Read Status` in asynchronous modes of operation.

CellularRAM 1.0, 64Mb PSRAM, for example MT45W4MW16BFB-701_1us

You can program these devices using a **CRE** pin, or by software access. Whenever you program these devices through software access, using a sequence of two reads followed by two writes, ensure that the third access, that is, the first write is a CE# controlled write.

The SMC only does WE# controlled writes. This is to simplify the design by having fewer timing registers and simpler timing controls.

Therefore, you can only program these devices using the **CRE** pin method of access.

1.3 Product revisions

This section describes differences in functionality between product revisions:

r1p0-r1p1 Contains the following differences in functionality:

- periph_id_2 Register bits [7:4] now read back as `0x2`. See *Peripheral Identification Register 2* on page 3-34.

r1p1-r1p2 Contains the following differences in functionality:

- periph_id_2 Register bits [7:4] now read back as `0x3`. See *Peripheral Identification Register 2* on page 3-34.
- Addition of the refresh_period_0 and refresh_period_1 registers. These enable the SMC to insert an idle cycle during consecutive bursts, that enables a PSRAM device to initiate a refresh cycle.

r1p2 - r2p0 Contains the following differences in functionality:

- periph_id _2 Register bits [7:4] now read back as `0x4`. See *Peripheral Identification Register 2* on page 3-34.
- Addition of optional SLC ECC support for NAND interfaces.

r2p0 - r2p1 Contains the following differences in functionality:

- periph_id _2 Register bits [7:4] now read back as `0x5`. See *Peripheral Identification Register 2* on page 3-34.
- Register bit assignments updated in the *NAND Cycles Register* on page 3-21. This enables the SMC to support a wider range of memory clock frequencies.
- For asynchronous multiplexed transfers, the assertion of **we_n** is programmable. See *SRAM Cycles Register* on page 3-20.
- Additional functionality added for the t_{RR} parameter. See *Data-to-command phase access timing* on page 2-40.

r2p1 - r2p2 Contains the following differences in functionality:

- periph_id _2 Register bits [7:4] now read back as `0x6`. See *Peripheral Identification Register 2* on page 3-34.
- The SMC now stores ECC codes to enable it to read erased pages and correctly identify that they contain no errors.

Chapter 2
Functional Description

This chapter describes the SMC operation. It contains the following sections:

2.1 Functional overview

Figure 2-1 shows an SMC block diagram.

Note

Depending on the configuration, you can implement either one or two memory interfaces and associated clock domains.

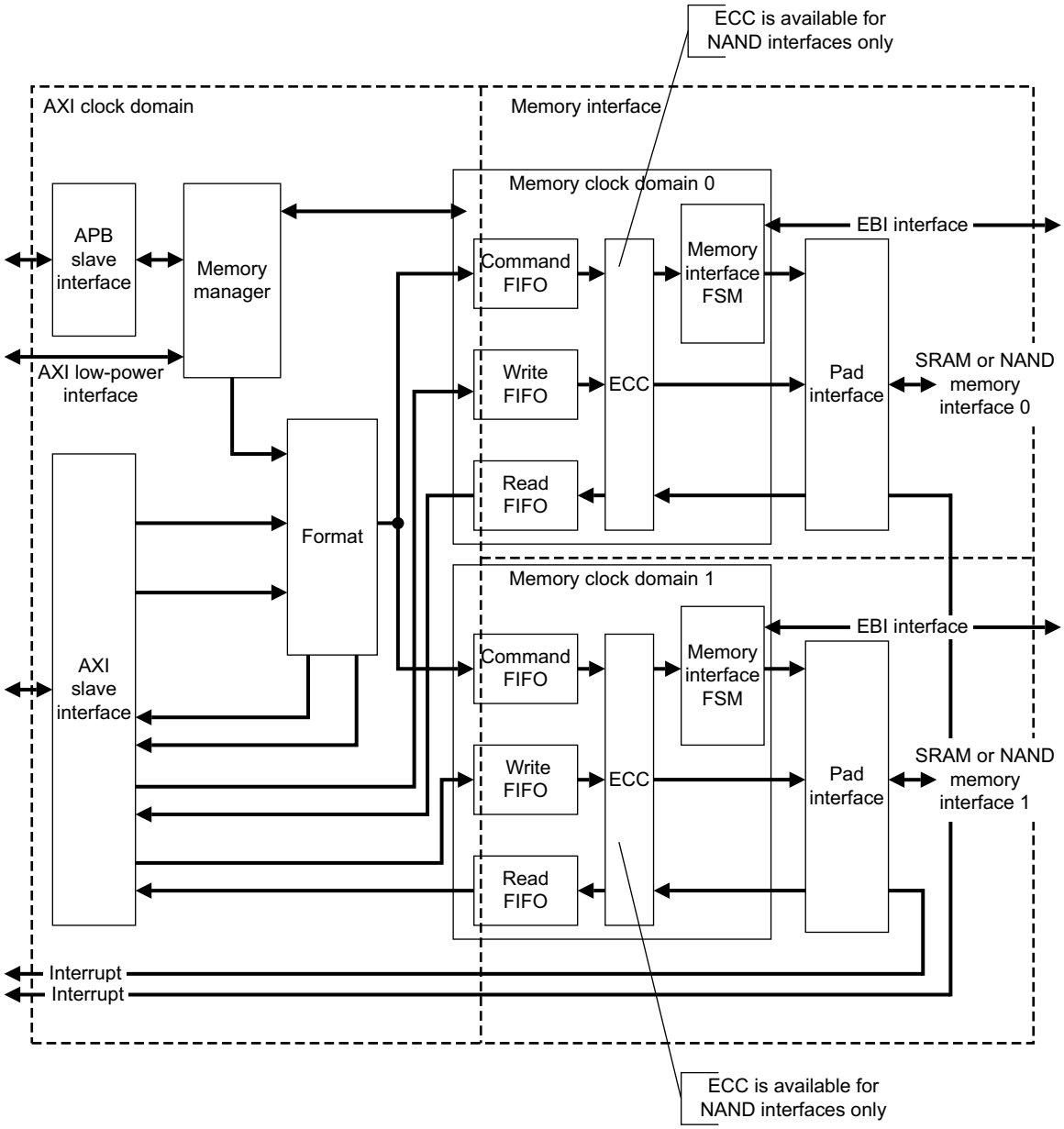


Figure 2-1 SMC block diagram

The main blocks of the SMC are:

- *AXI slave interface* on page 2-3
- *APB slave interface* on page 2-4
- *Format* on page 2-5
- *Memory manager* on page 2-6

- *Memory interface* on page 2-6
- *Pad interface* on page 2-6.

2.1.1 AXI slave interface

The AXI slave interface comprises five AXI channels:

- Read-Address (AR)
- Write-Address (AW)
- Write (W)
- Read (R)
- Write response (B).

For information on the AXI protocol, see the *AMBA AXI Protocol Specification*.

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

Note

In Figure 2-2 on page 2-4:

- The **arcache**, **awcache**, **arprot** and **awprot** signals are shown for completeness only. The SMC ignores any information that these signals provide.
- The clock and reset signals are not shown, see Table B-1 on page B-2.
- See Table B-11 on page B-6 for information about **wdata[PORTWIDTH–1:0]** and **wstrb[PORTBYTES–1:0]**.
- See Table B-14 on page B-9 for information about **rdata[PORTWIDTH–1:0]**.

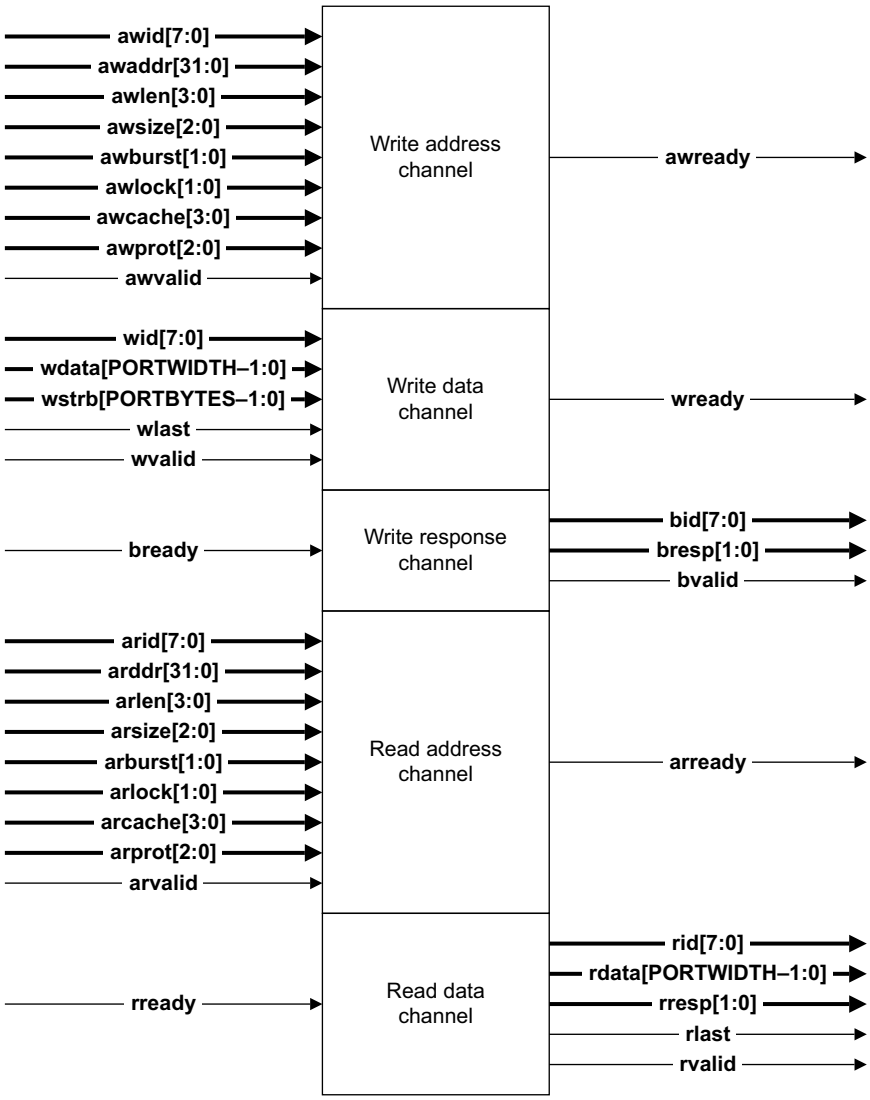


Figure 2-2 AXI slave interface signals

2.1.2 APB slave interface

Figure 2-3 shows the APB interface signals.

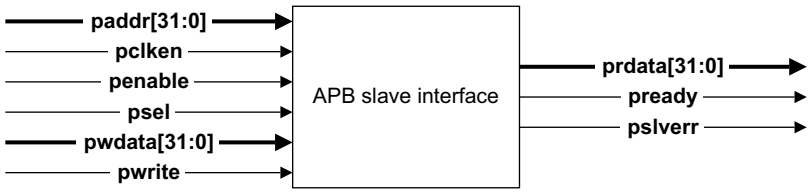


Figure 2-3 APB slave interface

Note

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

See *APB slave interface operation* on page 2-11 for more information.

2.1.3 Format

The format block receives memory accesses from the AXI slave interface and the memory manager. Requests from AR and AW channels are arbitrated on a round-robin basis. Requests from the manager have the highest priority. The format block also maps AXI transfers onto appropriate memory transfers and passes these to the memory interface through the command FIFO.

See *Format block* on page 2-11 for more information.

2.1.4 Memory manager

The memory manager tracks and controls the current state of the SMC **aclk** domain FSM. The block is responsible for:

- Updating register values that are used in the **mclk<x>** domain, and controlling direct commands issued to memory.
- Controlling entry-to and exit-from Low-power state through the APB interface.
- The AXI low-power interface. See *AXI low-power operation* on page 2-22.

See *Memory manager operation* on page 2-22 for more information.

2.1.5 Memory interface

The SMC supports two memory interface types, SRAM and NAND. Both SRAM and NAND memory interfaces are composed of command, read data, and write data FIFOs, and a control FSM. The memory interface FSM is specific to either SRAM or NAND. To support an EBI, the memory interface also contains an EBI FSM. This controls interaction with the EBI and prevents the memory interface FSM from issuing commands until it has been granted the external bus. NAND interfaces also have an optional *single-level cell* (SLC) ECC block.

See *Memory interface operation* on page 2-26 for more information.

2.1.6 Pad interface

The pad interface module provides a registered I/O interface for data and control signals. It also contains interrupt generation logic.

Figure 2-4 shows the SRAM interface, where x is the memory interface, 0 to 1, and n is the chip select, 0 to 3.

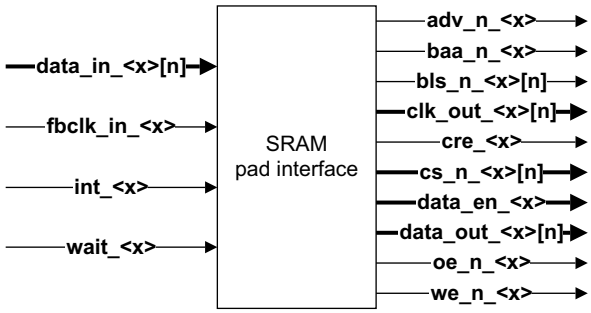


Figure 2-4 SRAM interface

Note

Figure 2-4 does not show the clock and reset signals.

Figure 2-5 on page 2-7 shows the NAND interface, where x is the memory interface, 0 to 1, and n is the chip select, 0 to 3.

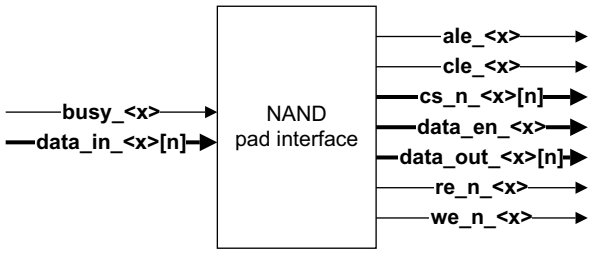


Figure 2-5 NAND interface

2.1.7 Interrupts

The SMC series provides interrupt outputs for use with NAND flash, because of the long wait times associated with this memory. Both the SRAM and NAND memory interface types support interrupts, and an interrupt is provided for each memory interface. The interrupt is triggered on the rising edge of:

- the **int** input for the SRAM memory interface type
- the **busy** input for the NAND memory interface type.

An additional interrupt is provided if an ECC block is included.

See *Interrupts operation* on page 2-25 for more information.

2.1.8 User signals

Figure 2-6 shows the user signals that the SMC provides.

Figure 2-6 User signals

See *Miscellaneous signals* on page 2-10 for more information.

2.2 Functional operation

This section describes:

- *Operating states*
- *Clocking and resets* on page 2-9
- *Miscellaneous signals* on page 2-10
- *APB slave interface operation* on page 2-11
- *Format block* on page 2-11
- *Memory manager operation* on page 2-22
- *Interrupts operation* on page 2-25
- *Memory interface operation* on page 2-26
- *SRAM interface timing diagrams* on page 2-26
- *NAND interface timing diagrams* on page 2-37
- *Error Correction Code block* on page 2-42.

2.2.1 Operating states

The operation of the SMC is based on three operating states. This section describes each of the states. Figure 2-7 shows the state machine.

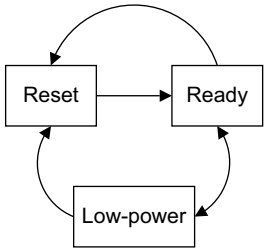


Figure 2-7 aclk domain FSM

The SMC states are as follows:

Reset Power is applied to the device, and **aresetn** is held LOW.

Ready Normal operation of the device. The APB interface can access the SMC register bank and the AXI interface can access external memory devices.

Low-power The device does not accept new AXI transfers. The APB interface can only access certain registers. You can stop the SMC clocks to reduce power consumption.

The state transitions are:

Ready to Reset When reset is asserted to the **aclk** domain, it enters the Reset state.

Reset to Ready When reset is deasserted to the **aclk** domain, it enters the Ready state.

Ready to Low-power

The SMC must enter the idle state before it can enter the Low-power state. The SMC enters Low-power state when:

- The low_power_req bit is set in the *Set Configuration Register* on page 3-11.
- The AXI master asserts **csysreq**. See *AXI low-power interface signals* on page B-9.

Note

After the SMC receives a low-power request, it does not respond to commands on the APB interface until it enters the Low-power state.

Low-power to Ready

The device exits the Low-power state back to Ready when either:

- The low_power_exit bit is set in the APB memc_cfg_clr Register. *Clear Configuration Register* on page 3-12.
- The AXI master deasserts **csysreq**. See *AXI low-power interface signals* on page B-9.

Low-power to Reset

When reset is asserted to the **aclk** reset domain, it enters the Reset state.

2.2.2 Clocking and resets

This section describes:

- *Clocking*
- *Reset* on page 2-10.

Clocking

All configurations of the SMC support at least two clock domains, and have the following clock inputs:

- **aclk**
- **mclk0**
- **mclk0n**.

The SMC-353 and SMC-354 configurations support two memory interfaces and therefore implement an additional clock domain and the following associated inputs:

- **mclk1**
- **mclk1n**.

These clocks can be grouped into three clock domains:

AXI domain **aclk** is in this domain. You can only stop **aclk** when the SMC is in Low-power state.

Memory clock domains

All the clocks except **aclk** are in these domains. Ensure that the **mclk<x>** signal is clocked at the rate of the external memory clock speed. **mclk<x>n** is an inverted version of **mclk<x>**. In configurations implementing two memory interfaces, the two memory clock domains can additionally be asynchronous to each other. You can stop the clocks for the memory clock domain when the SMC is in Low-power state.

Note

The **<x>** notation represents memory interface 0 or 1.

See the *CoreLink SMC-35x AXI Static Memory Controller Series Integration Manual* for the required relationships between the clocks.

You can tie-off the SMC **async<x>** and **msync<x>** pins so that the **aclk** and **mclk<x>** clock domains can operate synchronously or asynchronously with respect to each other. When you use the EBI (PL220), you must operate the SMC-353 and SMC-354 **mclk0** and **mclk1** clock domains synchronously at 1:1, n:1, or 1:n.

Synchronous clocking

The benefit of synchronous clocking is that you can reduce the read and write latency by removing the synchronization registers between clock domains. However, because of the integer relationship of the clocks, you might not be able to get the maximum performance from the system because of constraints placed on the bus frequency by the external memory clock speed.

In synchronous mode, the handshaking between the **aclk** and **mclk<x>** domains enables synchronous operation of the two clocks at multiples of each other, that is, ratios of n:1 and 1:m. Synchronous operation of the clocks can be 1:1, n:1, or 1:n.

Asynchronous clocking

The main benefit of asynchronous clocking is that you can maximize the system performance, while running the memory interface at a fixed system frequency. Additionally, in sleep-mode situations when the system is not required to do much work, you can lower the frequency to reduce power consumption.

Output clocks

A clock output is provided for every external memory device.

Reset

The SMC has up to three reset inputs:

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

mreset0n This is the reset signal for the **mclk0** domain.

mreset1n If a second memory interface is included, this is the reset signal for the **mclk1** domain.

You can change both reset signals asynchronously to their respective clock domain. Internally to the SMC, the deassertion of the **aresetn** signal is synchronized to **aclk**. The deassertion of **mreset0n** is synchronized internally to **mclk0** and **mclk0n**, and similarly, **mreset1n** is synchronized to **mclk1** and **mclk1n**.

2.2.3 Miscellaneous signals

You can use the following signals as general-purpose control signals for logic external to the SMC:

user_config[7:0] General purpose output signals that the write-only APB register drives directly. If you do not require these signals, leave them unconnected. See *User Config Register* on page 3-24.

user_status[7:0] General purpose input signals that are readable from the APB interface through the user_status Register. If you do not require these signals then tie them either HIGH or LOW. These signals are connected directly to the APB interface block. Therefore, if they are driven from external logic that is not clocked by the SMC **aclk** signal, then you require external synchronization registers. See *User Status Register* on page 3-23.

You can use the following miscellaneous signals as tie-offs to change the operational behavior of the SMC:

a_gt_m<x>_sync

When HIGH, it indicates that **aclk** is faster than, and synchronous to, **mclk<x>**.

async<x> When HIGH, it indicates **aclk** is synchronous to **mclk<x>**. Otherwise, they are asynchronous. Ensure that **async<x>** is tied to the same value as **msync<x>**.

dft_en_clk_out

Use this signal for *Automatic Test Pattern Generator* (ATPG) testing only. Tie it LOW for normal operation.

msync<x> When HIGH, indicates **mclk<x>** is synchronous to **aclk**. Otherwise, they are asynchronous. Ensure that **msync<x>** is tied to the same value as **async<x>**.

rst_bypass Use this signal for ATPG testing only. Tie it LOW for normal operation.

use_ebi When HIGH, it indicates that the SMC must operate with an EBI. See the *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual*.

2.2.4 APB slave interface operation

The APB interface is a fully compliant APB slave. The SMC has 4KB of memory allocated to it. For information describing the APB interface see the *AMBA 3 APB Protocol v1.0 Specification*.

The APB slave interface accesses the SMC registers to program the memory system configuration parameters and to provide status information. See Chapter 3 *Programmers Model* for more information.

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

To enable a clean registered interface to the external infrastructure the APB interface, always adds a wait state for all reads and writes by driving **pready** LOW during the first cycle of the access phase.

Note

The *AMBA 3 APB Protocol v1.0 Specification* defines access phase.

In two 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 an APB access is received, and a previous direct command has not completed.

2.2.5 Format block

This section describes:

Hazard handling

The following types of hazard exist:

- Read after read (RAR)
- Write after write (WAW)
- Read after write (RAW)
- Write after read (WAR).

The AXI specification defines that RAW and WAR ordering is determined by the master, whereas RAR and WAW ordering is enforced by the slave. If an AXI master requires ordering between reads and writes to certain memory locations, it must wait for a write response before issuing a read from a location it has written to (RAW). It must also wait for read data before issuing a write to a location it has read from (WAR). The SMC ensures the ordering of read transfers from a single master is maintained (RAR), and additionally, that the ordering of write transfers from a single master is maintained (WAW).

RAR

RAR hazards only occur in configurations that have two memory interfaces.

The SMC can reorder reads from different masters that connect to different memory interfaces. This situation is likely to occur, for example, in an SMC-353 configuration, when one master is accessing an SRAM memory interface clocked at 133MHz, and another master is accessing the NAND memory interface clocked at 50MHz. Read data from the SRAM memory is available before data from the NAND memory. This enables the SMC to potentially return read data out of order. The SMC contains internal hazard checking to ensure the AXI reads from a single master have the order maintained.

WAW

WAW hazards only occur in configurations that have two memory interfaces.

As for RAR hazards, writes to different memory interfaces are able to complete out of order. This enables the write responses to be returned out of order. The SMC internal hazard checking logic ensures only writes from different masters are completed out of order.

Exclusive accesses

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 accesses

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 shows 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`
		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 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`.

NAND memory accesses

This section describes:

- *Two phase NAND accesses*
- *NAND command phase transfers* on page 2-15
- *NAND data phase transfers* on page 2-16.

Two phase NAND accesses

The SMC defines two phases of commands when transferring data to or from NAND flash.

Command phase
: Commands and optional address information are written to the NAND flash. The command and address can be associated with either a data phase operation to write to or read from the array, or a status/ID register transfer.

Data phase
: Data is either written to or read from the NAND flash. This data can be either data transferred to or from the array, or status/ID register information.

The SMC uses information contained in the AXI address bus, either **awaddr[ ]** or **araddr[ ]** signals, to determine whether the AXI transfer is a command or data phase access.

This information contained in the address bus additionally determines:

- the value of the command
- the number of address cycles
- the chip select to be accessed.

During a command phase transfer, the address to be written to the NAND memory is transferred to the SMC using the AXI write channel.

Note

The size of the AXI transfer for data phase transfers must be larger than the width of the memory interface.

Table 2-2 shows the fields of **awaddr[ ]** and **araddr[ ]** signals that control a NAND flash transfer.

Table 2-2 NAND AXI address setup

AXI address	Command phase	Data phase
[31:24]	Chip address	Chip address
[23]	NoOfAddCycles_2	Reserved
[22]	NoOfAddCycles_1	Reserved
[21]	NoOfAddCycles_0	ClearCS
[20]	End command valid	End command valid[a]
[19]	0	1
[18:11]	End command	End command[b]
[10:3]	Start command	[10] ECC Last [9:3] Reserved
[2:0]	Reserved[c]	Reserved[c]

a. For a read data phase transaction, the end command valid must be 0.
b. End command data is ignored if end command valid is not true.
c. The bottom three bits of a NAND access determine the valid data byte lanes, in the same way as for a standard AXI access.

NAND command phase transfers

A command phase transfer is always performed as an AXI write. The AXI **awaddr[ ]** bus, and Table 2-2 on page 2-14 contain the following information:

Address cycles

The number of address cycles can be any value from zero to seven. Generally, up to five cycles are used during an array read or write, but a maximum of seven enables support for future devices.

Start command

The start command is used to initiate the required operation, for example:

- page read
- page program
- random page read
- status or ID register read.

End command

The value of the second command, if required. This command is executed when all address cycles have completed. For example, some NAND memories require an additional command, following the address cycles, for a page read.

End command valid

Indicates whether the end command must be issued to the NAND flash.

Each address cycle consumes eight bits of address information. This is transferred to the SMC through the AXI write channel.

Note

To ease system integration, the SMC supports the use of multiple AXI write transactions to transfer address information. The following restrictions apply in this case:

1. The AXI address [31:3] bits must not change between transactions. The first transaction must be doubleword aligned.
2. All other address information must be the same, with the exception of transaction length.
3. Data must be transferred in incrementing, consecutive accesses, that is, not wrapping, fixed, or sparse.
4. Extra or unused beats in the last transaction must have write strobes disabled.
5. Total number of beats must be less than the write FIFO depth.

NAND data phase transfers

Transfers data to or from the NAND flash, and can be performed as either an AXI read or write, depending on the required operation. The **araddr[ ]** or **awaddr[ ]** bus, and Table 2-2 on page 2-14 contain this information:

End command

The value of a command that is issued following the data transfer. This is required by some memories to indicate a page program following input of write data.

Note

End commands are not supported for read data phase transfers.

End command valid

Indicates whether the end command must be issued to NAND flash.

ClearCS When set, the chip select for a NAND flash is deasserted on completion of this command. When not set, the chip select remains asserted.

ECC Last When set, this bit indicates to the ECC block that the current command is the last access to the NAND page. It is ignored if the ECC block is not enabled. See *Error Correction Code block* on page 2-42.

A NAND flash data phase program or read operation is expected to require multiple AXI transfers because of the large page size of NAND memories. Some memory devices require the chip select to remain asserted for the duration of a page access. The SMC keeps the chip select asserted for the:

Command phase to data phase transition

When **nand_csl** is tied HIGH.

Data phase to data phase transitions

When the ClearCS is not set. The SMC asserts the chip select while multiple AXI transfers transfer data to or from the NAND flash internal page. On the last AXI transfer, you can deassert the chip select by setting the ClearCS bit.

Note

Using the optional ClearCS functionality or the **nand_csl** signal causes the SMC to keep requesting the EBI, until the chip select is cleared. You must take care, at the system level, to ensure that this does not cause a deadlock when using the EBI.

Figure 2-8 on page 2-17 and Figure 2-9 on page 2-18 show the steps taken to perform NAND flash page read and page program operations respectively.

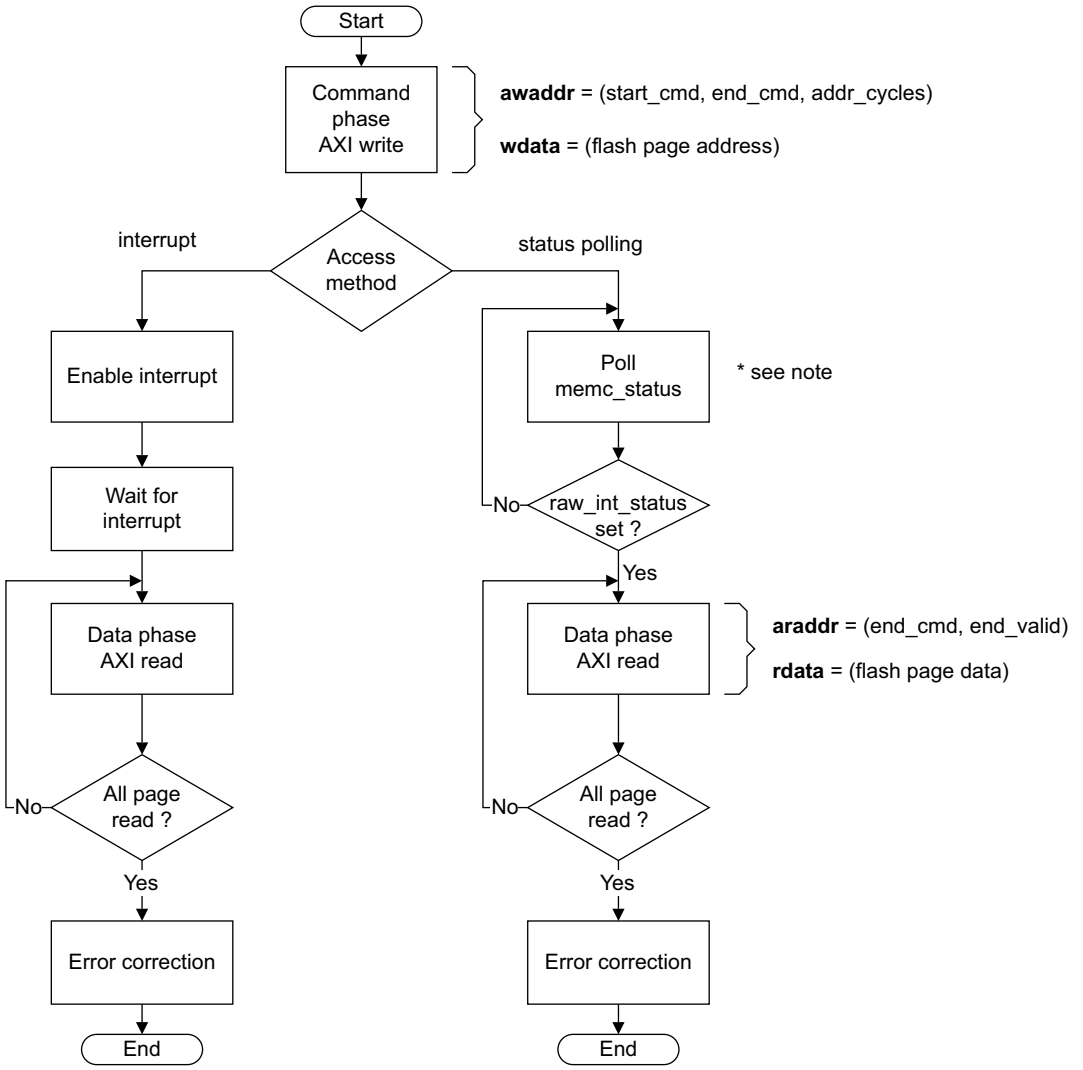


Figure 2-8 NAND flash page read operations

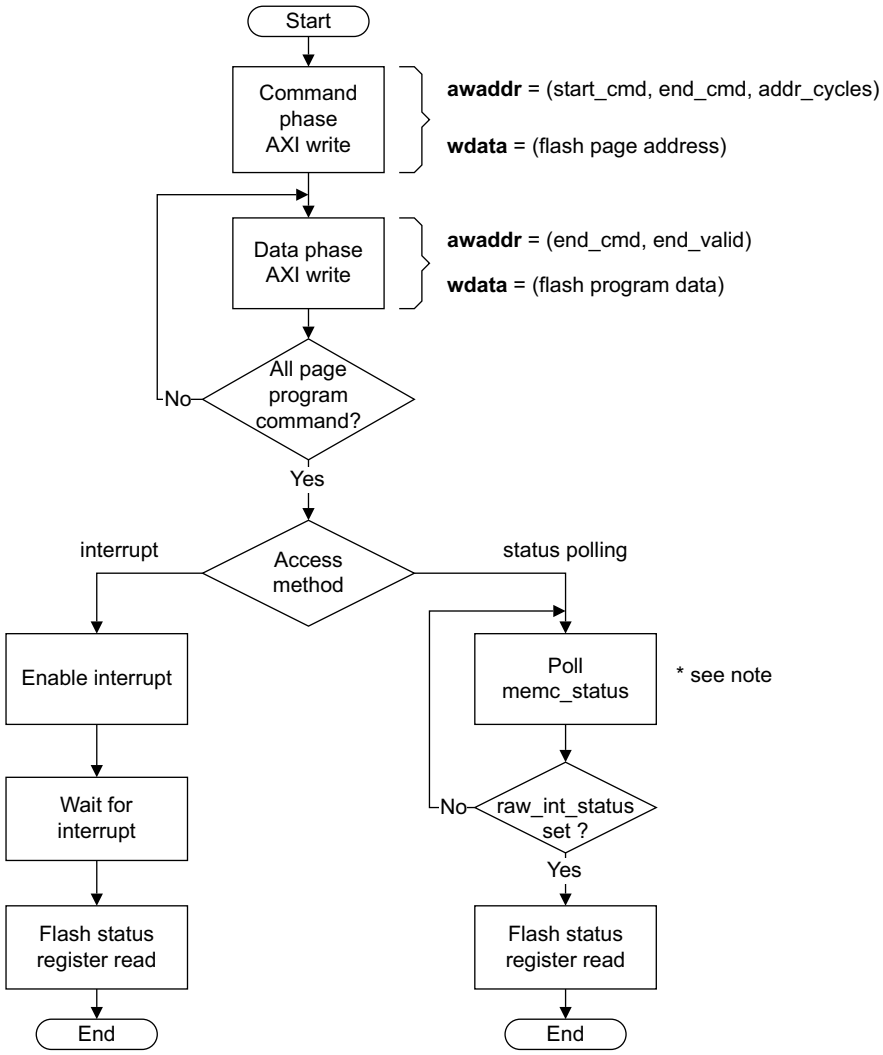


Figure 2-9 NAND flash page program operations

Note

You can poll for either a page program or page read completion in two ways:

- Poll the raw_int_status bit in the memc_status Register to determine when the memory **busy_<x>** output has gone HIGH, indicating a page program completion or read data ready.
- In a system with multiple NAND flash devices connected to the SMC. The **busy** outputs are wire-ANDed to produce the single **busy_<x>** input to the SMC, that only transitions HIGH when all devices have completed. You can determine the status register of each NAND chip by reading the individual device status register.

Figure 2-10 on page 2-19 shows the steps taken to perform a NAND flash status register read.

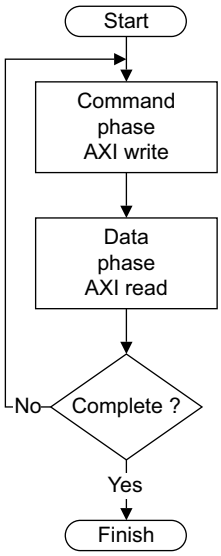


Figure 2-10 NAND flash status register read

The process boxes that Figure 2-10 shows are defined as:

Command phase AXI write

awaddr = (`start_cmd = STATUS_READ_CMD`,
`end_cmd = 0x0`
`end_valid = 0x0`
`addr_cycles = 0x0`)

Note

Ensure burst length is 1, **awlen** = `0x0`.

wdata = (don't care).

Data phase AXI read

araddr = (`ClearCS` = `0x1`).

Note

Ensure burst length is 1, **arlen** = `0x0`.

The transfer size is eight or sixteen, **arsize** = `0x0` or `0x1`.

rdata = NAND flash status output.

Note

Certain NAND flash devices can support multiple status register reads without reissuing the STATUS_READ_CMD. In this case, you can modify the flow that *NAND flash status register read* describes to include multiple data phase transfers for each command phase transfer.

The upper byte of the address read or write bus, **araddr[31:24]** or **awaddr[31:24]**, and the value of the **address_match[]** and **address_mask[]** buses determine the chip select being accessed. To select a memory device either:

- **araddr[31:24]** & **address_mask[]** must equal **address_match[]**
- **awaddr[31:24]** & **address_mask[]** must equal **address_match[]**

The values for the **address_mask** and **address_match** buses must be set so that no address maps onto more than one chip, otherwise the behavior of the SMC is undefined. If an AXI access does not map to any memory device then the SMC performs an asynchronous transfer on memory interface 0 with all chip selects deasserted. After the transfer completes, the SMC provides an OKAY response.

SRAM memory accesses

This section describes:

- *Standard SRAM accesses*
- *Memory address shifting*
- *Memory burst alignment* on page 2-21
- *Memory burst length* on page 2-21
- *Booting using the SRAM interface* on page 2-21.

Standard SRAM accesses

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

The upper byte of the address read or write bus, **araddr[31:24]** or **awaddr[31:24]**, and the value of the **address_match[]** and **address_mask[]** buses determine the chip select being accessed. To select a memory device either:

- **araddr[31:24]** & **address_mask[]** must equal **address_match[]**
- **awaddr[31:24]** & **address_mask[]** must equal **address_match[]**

The values for the **address_mask** and **address_match** buses must be set so that no address maps onto more than one chip, otherwise the behavior of the SMC is undefined. If an AXI access does not map to any memory device then the SMC performs an asynchronous transfer on memory interface 0 with all chip selects deasserted. After the transfer completes, the SMC provides an OKAY response.

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.

Note

The **arcache**, **awcache**, **arprot** and **awprot** signals are included in the AXI interface list for completeness only. The SMC does not use the information transferred by these signals.

Memory address shifting

To produce the address presented to the memory device, the AXI address is aligned to the memory width. This is done because the AXI address is a byte-aligned address, whereas the memory address is a memory-width-aligned address.

Note

During initial configuration of a memory device, the memory mode register can be accessed with a sequence of transfers to specific addresses. You must take into consideration the shifting performance by the SMC when accessing memory mode registers.

Memory burst alignment

The SMC provides a programmable option for controlling the formatting of memory transfers with respect to memory burst boundaries, through the burst_align bit of the opmode registers.

When set, the burst_align bit causes memory bursts to be aligned to a memory burst boundary. This setting is intended for use with memories that use the concept of internal pages. This can be an asynchronous page mode memory, or a synchronous PSRAM. If an AXI burst crosses a memory burst boundary, the SMC partitions the AXI transfer into multiple memory bursts, terminating a memory transfer at the burst boundary. Ensure the page size is an integer multiple of the burst length, to avoid a memory burst crossing a page boundary.

When the burst_align bit is not set, the SMC ignores the memory burst boundary when mapping AXI commands onto memory commands. This setting is intended for use with devices such as NOR flash. These devices have no concept of pages.

Memory burst length

The SMC enables you to program the memory burst length on an individual chip basis, from length 1 to 32 beats, or a continuous burst. The length of memory bursts are however automatically limited by the size of the read or write data FIFOs.

For read transfers, the maximum memory burst length on the memory interface is the depth of the read data FIFO. For writes, the maximum burst length is dependent on:

- the beat size of the AXI transfer, `asize`
- the memory data bus width, `mw`
- the depth of the write data FIFO depth, `wfifo_depth`.

The formula to determine the maximum memory write burst length is:

Memory write burst length = `((1<<asize) x wfifo_depth) / (1<<mw)`

Booting using the SRAM interface

The SMC enables the lowest SRAM chip select, normally chip 0, to be bootable. To enable SRAM memory to be bootable, the SRAM interface does not require any special functionality, other than knowing the memory width of the memory concerned. This is indicated by a top-level tie-off. To enable the SMC to work with the slowest memories, the timing registers reset to the worst-case values. When the **remap_<x>** signal is HIGH, the memory with the bootable chip select is set by the **sram_mw_<x>[1:0]** tie-off signals.

Additionally, while the SMC input **remap_<x>** is HIGH, the bootable chip is aliased to base address `0x0`.

2.2.6 Memory manager operation

The memory manager module is responsible for controlling the state of the SMC and updating the chip configuration registers.

This subsection describes:

- *AXI low-power operation*
- *Chip configuration registers*
- *Direct commands* on page 2-23.

AXI low-power operation

The SMC accepts requests to enter the Low-power state through either the AXI low-power interface, or the APB register interface.

The SMC does not enter the power-down state until it has received an idle indication from all areas of the peripheral, that is:

- there is no valid transfer held in the format block
- there are no valid transfers held in the AXI interface
- all FIFOs are empty
- all memory interface blocks are IDLE.

When the Low-power state is entered, the AXI outputs **awready**, **arready**, and **wready** are driven LOW to prevent any new AXI transfers being accepted. No new AXI transfers are accepted until the SMC has been moved out of Low-power state. The SMC does not request to move out of Low-power state, and never refuses a power-down request.

Chip configuration registers

The SMC provides a mechanism for synchronizing the switching of operating modes with that of the memory device.

The *Set Cycles Register* on page 3-15 and *Set Operating Mode Register* on page 3-16 act as holding registers for new operating parameters until the SMC detects the memory device has switched modes. This enables a memory device to be made to change its operating mode while still being accessed.

Figure 2-11 on page 2-23 shows the memory manager containing a bank of registers for each memory chip supported by the SMC configuration. The manager register bank consists of all the timing parameters `chip<x>_cycles`, and access modes `chip<x>_opmode`. These are required for the SMC to correctly time any type of access to a supported memory type.

The APB registers set_cycles and set_opmode act as holding registers, the configuration registers within the manager are only updated if either:

- the *Direct Command Register* on page 3-13 indicates only a register update is taking place
- the direct_cmd Register indicates a mode register access either using the direct_cmd Register or using the AXI interface and the command has completed.

The chip configuration registers are available as read-only registers in the address map of the APB interface.

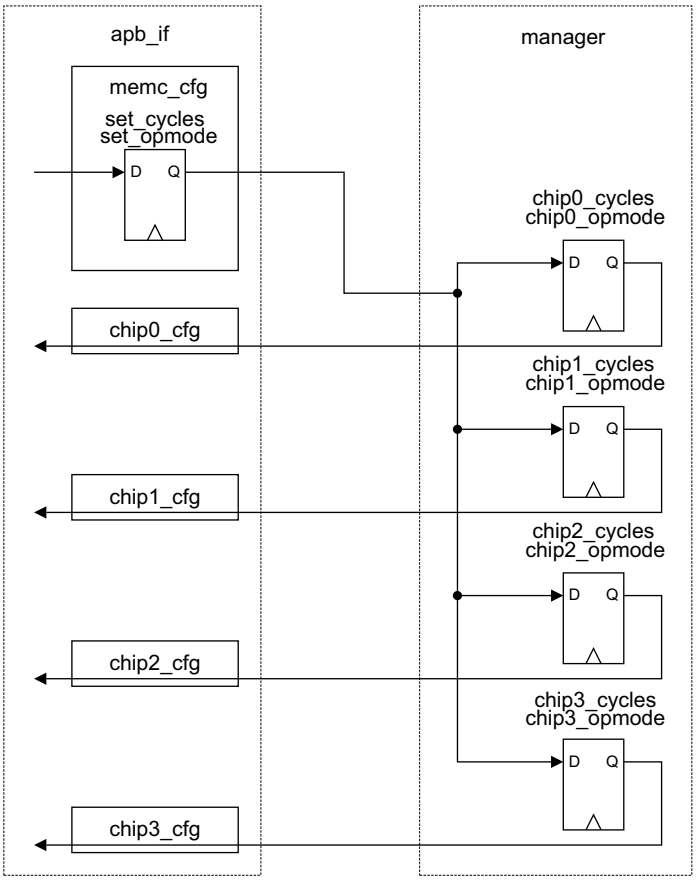


Figure 2-11 Chip configuration registers

Direct commands

The SMC enables code to be executed from the memory while simultaneously, from the software perspective, moving the same chip to a different operating mode. This is achieved by synchronizing the update of the chip configuration registers from the holding registers with the dispatch of the memory configuration register write.

The SMC provides two mechanisms for simultaneously updating the controller and memory configuration registers. These are:

Device pin mechanism

For memories that use an input pin to indicate that a write is intended for the configuration register, for example some PSRAM devices, the write mechanism can be implemented using the *Direct Command Register* on page 3-13. Figure 2-12 on page 2-24 shows the sequence of events.

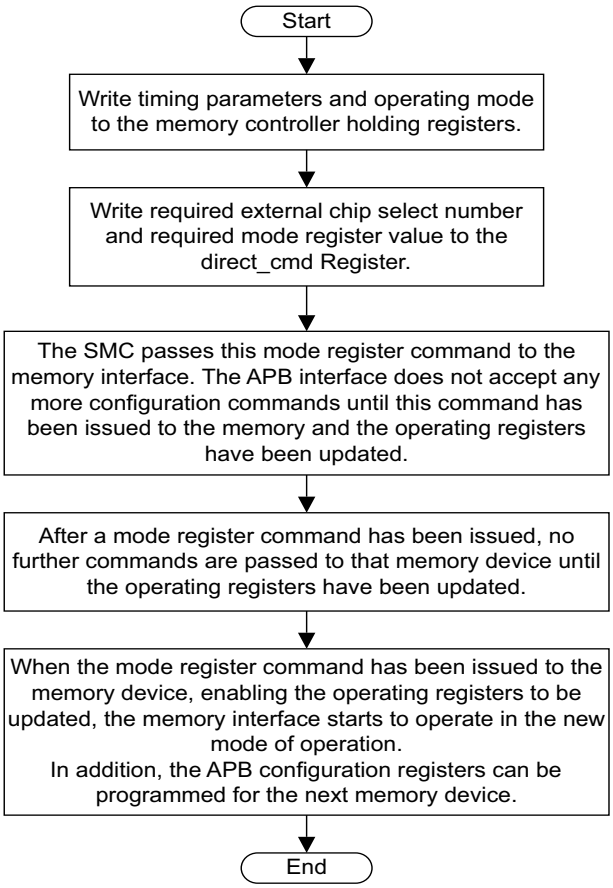


Figure 2-12 Device pin mechanism

Software mechanism

For memories that require a sequence of read and write commands, for example, most NOR flash devices use the AXI interface, with the write data bus used to indicate when the last transfer has completed and when it is safe for the SMC to update the chip configuration registers. Figure 2-13 on page 2-25 shows the sequence of events.

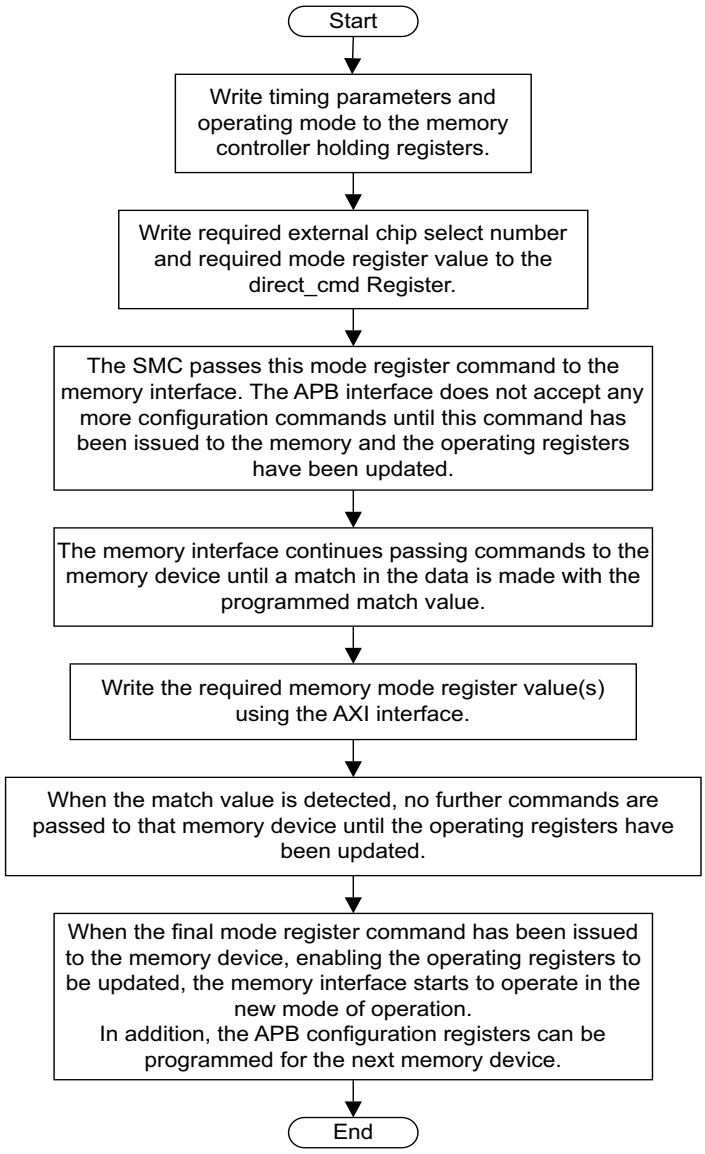


Figure 2-13 Software mechanism

2.2.7 Interrupts operation

The busy outputs of each chip are wire-ANDed together, external to the SMC, to create a single **busy_<x>** (NAND interfaces) or **int_<x>** (SRAM interfaces) signal. This signal creates an internal interrupt input per memory interface. Multiple outstanding accesses to NAND chips only trigger an interrupt when all chips have completed the respective operations. During the busy phase, you can read the status register of each chip to determine which chips have completed.

An interrupt is cleared by the next AXI read to any chip select on the appropriate memory interface, or by a write to the appropriate bit in the *Clear Configuration Register* on page 3-12.

The interrupt outputs are generated through a combinational path from the relevant input pin. This enables the SMC to be placed in Low-power state, and the clocks stopped, while waiting for an interrupt.

When interrupts are disabled, a synchronized version of the interrupt input is still readable through the APB interface. This enables software to poll, rather than use an interrupt to determine when NAND operations can proceed. There is also an interrupt for each ECC block. See *Error Correction Code block* on page 2-42.

2.2.8 Memory interface operation

The memory interface issues commands to the memory from the command FIFO, and controls the cycle timings of these commands. It only issues a new command after the previous command is complete and any turn-around times have been met. It only issues a read command when there is space for all the impending data in the read data FIFO.

Note

- The rd_bl field in the *Operating Mode Status Register* on page 3-21 must not be set greater than the read data FIFO depth.
- The SMC does not perform WRAP transfers on the memory interface. For memory devices that only operate in WRAP mode, you must program the *Set Operating Mode Register* on page 3-16 to align transfers to a memory burst boundary. If the SMC is programmed to perform transfers that cross a memory boundary, then you must program the memory device to operate in INCR mode.

If enabled, the EBI can prevent commands being issued when the SMC is not granted the external bus.

2.2.9 SRAM interface timing diagrams

All address, control, and write data outputs of the SMC are registered on the rising edge of **mclk<x>n**, equivalent to the falling edge of **mclk<x>**, for both synchronous and asynchronous accesses. The clock output to memory, **clk_out**, is driven directly by **mclk<x>**, but gated to prevent toggling during asynchronous accesses, or when no transfers are occurring.

Read data output by the memory device is also registered on the rising edge of **mclk<x>n**, equivalent to the falling edge of **mclk<x>**, for asynchronous reads. For synchronous reads, read data is registered using the fed-back clock, **fbclk_in**. For synchronous and asynchronous accesses, the data is then pushed onto the read data FIFO to be returned by the AXI interface.

Note

The internal signal **read_data** is included in the read transfer waveforms to indicate the clock edge on which data is registered by the SMC.

This subsection describes:

- *Asynchronous read* on page 2-27
- *Asynchronous read in multiplexed mode* on page 2-27
- *Asynchronous write* on page 2-28
- *Asynchronous write in multiplexed mode* on page 2-29
- *Asynchronous page mode read* on page 2-30
- *Synchronous burst read* on page 2-31
- *Synchronous burst read in multiplexed mode* on page 2-32
- *Synchronous burst write* on page 2-32
- *Synchronous burst write in multiplexed mode* on page 2-33
- *Synchronous read and asynchronous write* on page 2-34

- *Programming tRC and tWC when the controller operates in synchronous mode* on page 2-35
- *Chip select assertion for SRAM memory interfaces* on page 2-36.

Asynchronous read

Table 2-3 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-3 Asynchronous read opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	0	b000	-	-	-	-	-	-

Table 2-4 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-4 Asynchronous read sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	b0011	-	b001	-	-	-	-

Figure 2-14 shows a single asynchronous read transfer with an initial access time, t_{RC}, of 3 cycles and an output enable assertion delay, t_{CEOE}, of one cycle.

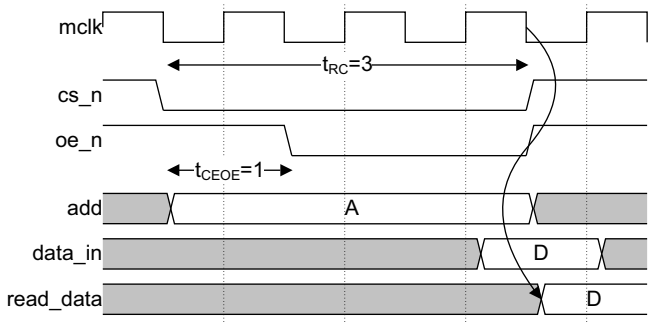


Figure 2-14 Asynchronous read

Asynchronous read in multiplexed mode

Table 2-5 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-5 Asynchronous read in multiplexed mode opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	0	b000	-	-	-	1	-	-

Table 2-6 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-6 Asynchronous read in multiplexed mode sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	b0111	-	b101	-	-	-	-

Figure 2-15 shows a single asynchronous read transfer in multiplexed SRAM mode, with t_{RC}=7, and t_{CEOE}=5.

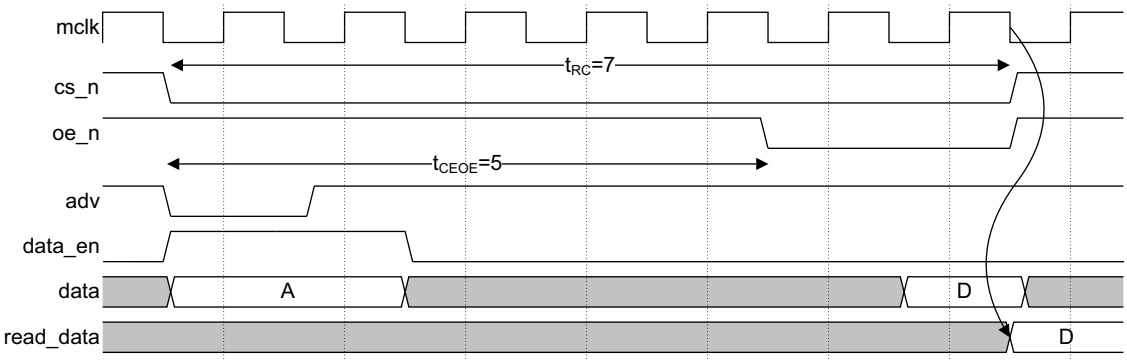


Figure 2-15 Asynchronous read in multiplexed mode

Note

In multiplexed mode, both address and data are output by the SMC on the **data_out** bus. Read data is accepted on the **data_in** bus. The address is still driven onto the address bus in multiplexed mode. This enables you to use the upper address bits for memories that require more address bits than data bits.

Asynchronous write

Table 2-7 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-7 Asynchronous write opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	-	-	0	b000	-	-	-	-

Table 2-8 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-8 Asynchronous write sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	-	b0100	-	b010	-	-	0

Figure 2-16 on page 2-29 shows an asynchronous write with a write cycle time t_{WC} of four cycles and a **we_n** assertion duration, t_{WP}, of two cycles.

Note

The timing parameter t_{WP} controls the deassertion of **we_n**. You can use it to vary the hold time of **cs_n**, **addr** and **data**. This differs from the read case where the timing parameter t_{CEOE} controls the delay in the assertion of **oe_n**. Additionally, **we_n** is always asserted one cycle after **cs_n** to ensure the address bus is valid.

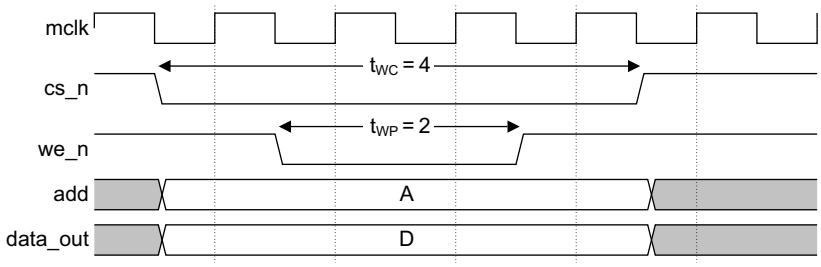


Figure 2-16 Asynchronous write

Asynchronous write in multiplexed mode

Table 2-9 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-9 Asynchronous write in multiplexed mode opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	-	-	0	b000	0	0	-	-

Table 2-10 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-10 Asynchronous write in multiplexed mode sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	-	b0111	-	b100	-	-	0, see Figure 2-17. 1, see Figure 2-18 on page 2-30.

Figure 2-17 shows an asynchronous write in multiplexed mode when the we_time bit is 0. t_{WC} is seven cycles, t_{WP} is four cycles, and the we_time bit programs the assertion of **we_n** to occur two clock cycles after **cs_n** goes LOW.

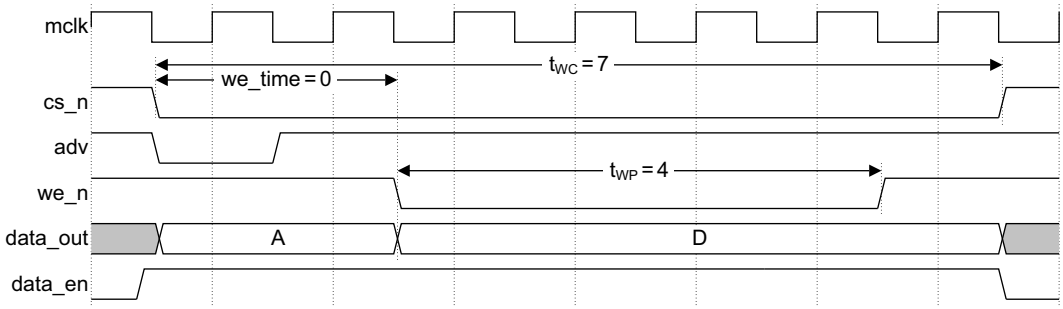


Figure 2-17 Asynchronous write in multiplexed mode when we_time is zero

Figure 2-18 on page 2-30 shows an asynchronous write in multiplexed mode when the we_time bit is 1. t_{WC} is seven cycles, t_{WP} is four cycles, and the we_time bit programs the assertion of **we_n** to occur when **cs_n** goes LOW.

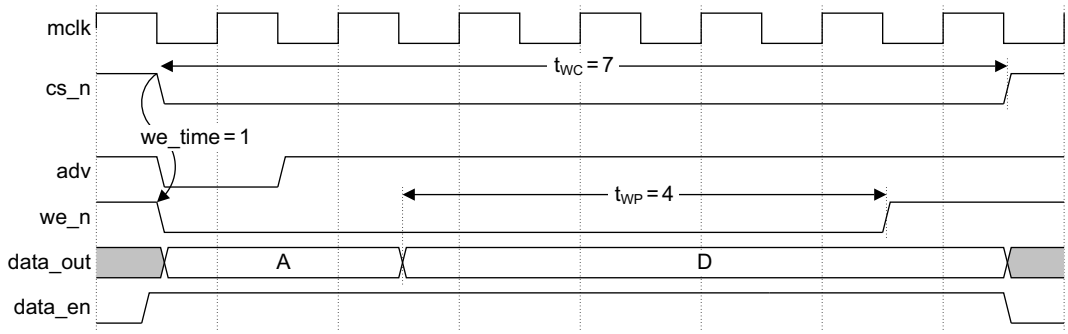


Figure 2-18 Asynchronous write in multiplexed mode when we_time is one

Asynchronous page mode read

Table 2-11 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-11 Page read opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	0	<page length>	-	-	-	-	-	1

Table 2-12 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-12 Page read sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	b0011	-	b010	-	b001	-	-

Figure 2-19 shows a page read access, with an initial access time, t_{RC}, of three cycles, an output enable assertion delay, t_{CEOE}, of two cycles, and a page access time, t_{PC}, of one cycle.

You enable Page mode in the SMC by setting the opmode Register for the relevant chip to asynchronous reads, and the burst length to the page size.

Note

Multiplexed mode page accesses are not supported.

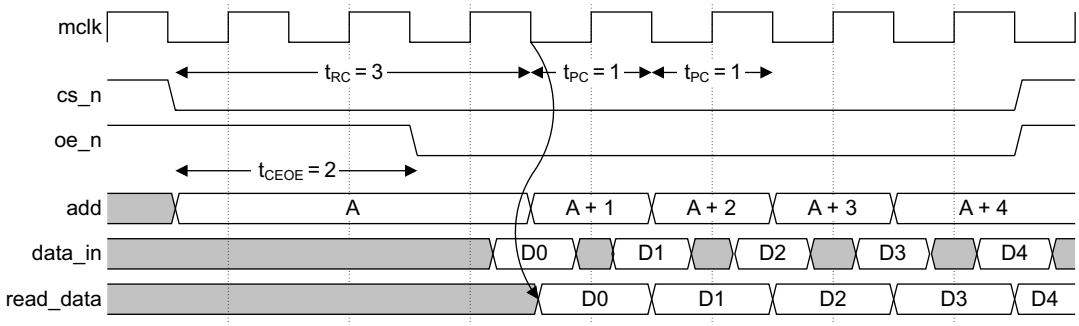


Figure 2-19 Page read

Synchronous burst read

Table 2-13 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-13 Synchronous burst read opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	1	<burst length>	-	-	-	1	-	-

Table 2-14 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-14 Synchronous burst read sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	b0100	-	b010	-	-	-	-

Figure 2-20 shows a burst read with the **wait** output of the memory used to delay the transfer.

Note

- Synchronous memories have a configuration register enabling **wait** to be asserted either on the same clock cycle as the delayed data, or a cycle early. The SMC only supports **wait** being asserted one cycle early, enabling **wait** to be initially sampled with the fed-back clock and then with **mclk** before being used by the FSM. This enables the easiest timing closure. Additionally, you must configure the memory for **wait** to be active LOW.
- In synchronous operation, the SMC relies on the **wait** signal being deasserted HIGH to indicate that the memory can finish the transfer. When in synchronous mode, some memories do not deassert the **wait** signal during non-array read transfers. Non-array read transfers are typically status register reads. To avoid stalling the system with these memories, in synchronous mode you must not perform non-array read transfers with the memory and SMC.
- You must set t_{RC} to a value that enables **wait_reg_mclk** to stabilize. See Figure 2-20.

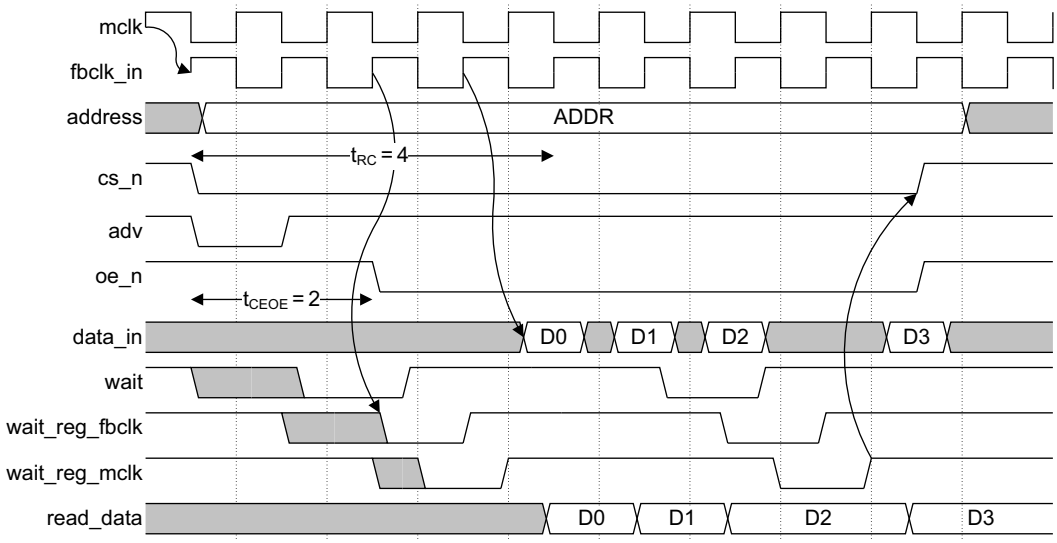


Figure 2-20 Synchronous burst read

Synchronous burst read in multiplexed mode

Table 2-15 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-15 Synchronous burst read in multiplexed mode opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	1	<burst length>	-	-	-	-	-	-

Table 2-16 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-16 Synchronous burst read in multiplexed mode read sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	b0100	-	b010	-	-	-	-

Figure 2-21 shows the same synchronous read burst transfer as Figure 2-20 on page 2-31, but in multiplexed mode.

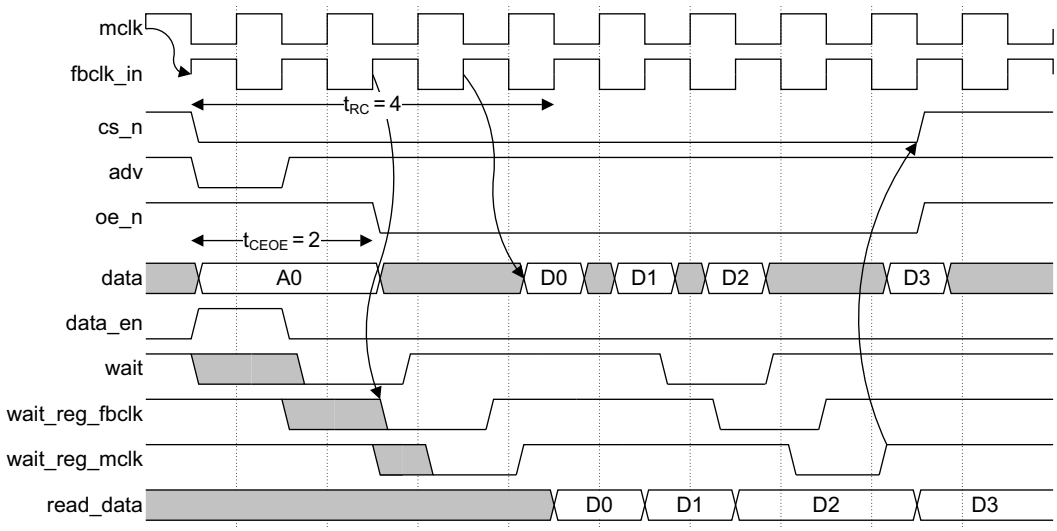


Figure 2-21 Synchronous burst read in multiplexed mode

Synchronous burst write

Table 2-17 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-17 Synchronous burst write opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	-	-	1	<burst length>	-	1	-	-

Table 2-18 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-18 Synchronous burst write sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	-	b0100	-	b001	-	-	-

Figure 2-22 shows a synchronous burst write transfer that is delayed by the **wait** signal. You must configure the memory to assert **wait** one cycle early and with an active LOW priority. The **wait** signal is again registered with the fed-back clock and **mclk** before being used. The **wait** signal is used in the **mclk** domain to the memory interface FSM.

Note

- Synchronous memories have a configuration register enabling **wait** to be asserted either on the same clock cycle as the delayed data, or a cycle early. The SMC only supports **wait** being asserted one cycle early, enabling **wait** to be initially sampled with the fed-back clock and then with **mclk** before being used by the FSM. This enables the easiest timing closure. Additionally, you must configure the memory for **wait** to be active LOW.
- You must set t_{WC} to a value that enables **wait_reg_mclk** to stabilize. See Figure 2-22.

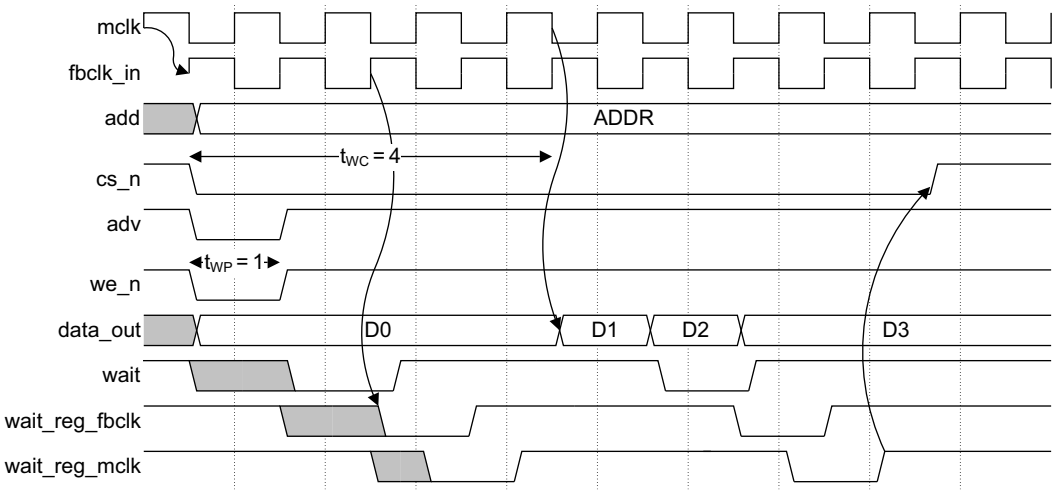


Figure 2-22 Synchronous burst write

Synchronous burst write in multiplexed mode

Table 2-19 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-19 Synchronous burst write in multiplexed mode opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	-	-	1	<burst length>	-	1	-	-

Table 2-20 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-20 Synchronous burst write in multiplexed mode sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	-	b0100	-	b001	-	-	-

Figure 2-23 on page 2-34 shows the same synchronous burst write as Figure 2-22, but in multiplexed mode.

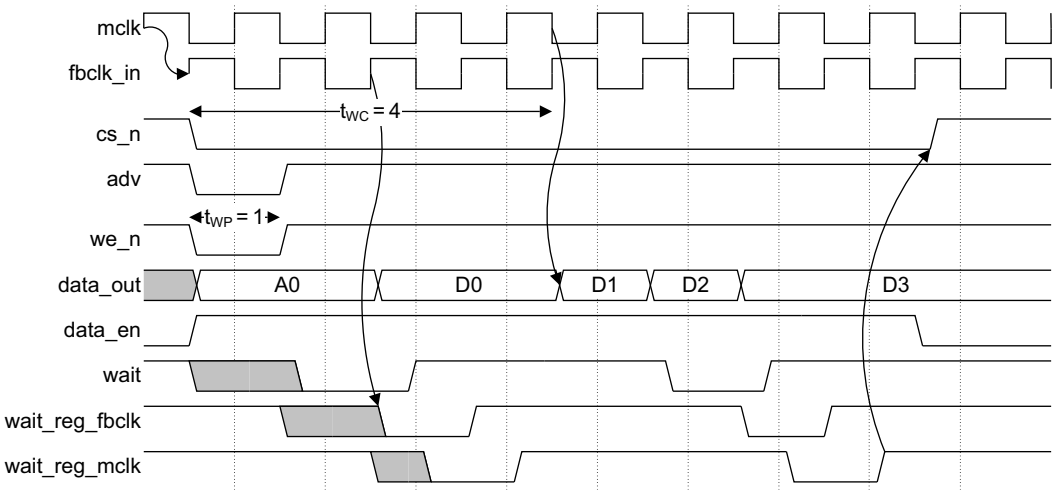


Figure 2-23 Synchronous burst write in multiplexed mode

Synchronous read and asynchronous write

Table 2-21 shows the settings for the *Operating Mode Status Register* on page 3-21.

Table 2-21 Synchronous read and asynchronous write opmode Register settings

Field	mw	rd_sync	rd_bl	wr_sync	wr_bl	baa	adv	bls	burst_align
Value	-	1	b001	0	b000	0	1	0	-

Table 2-20 on page 2-33 shows the settings for the *SRAM Cycles Register* on page 3-20.

Table 2-22 Synchronous read and asynchronous write sram_cycles Register settings

Field	t_rc	t_wc	t_ceoe	t_wp	t_pc	t_tr	we_time
Value	b0100	b0110	b010	b001	-	b011	-

Figure 2-24 on page 2-35 shows the turnaround time t_{TR}, enforced between synchronous read and asynchronous write. The turnaround time is enforced between:

- reads followed by writes
- writes followed by reads
- read following a read from a different chip select
- any two consecutive accesses in multiplexed address/data mode.

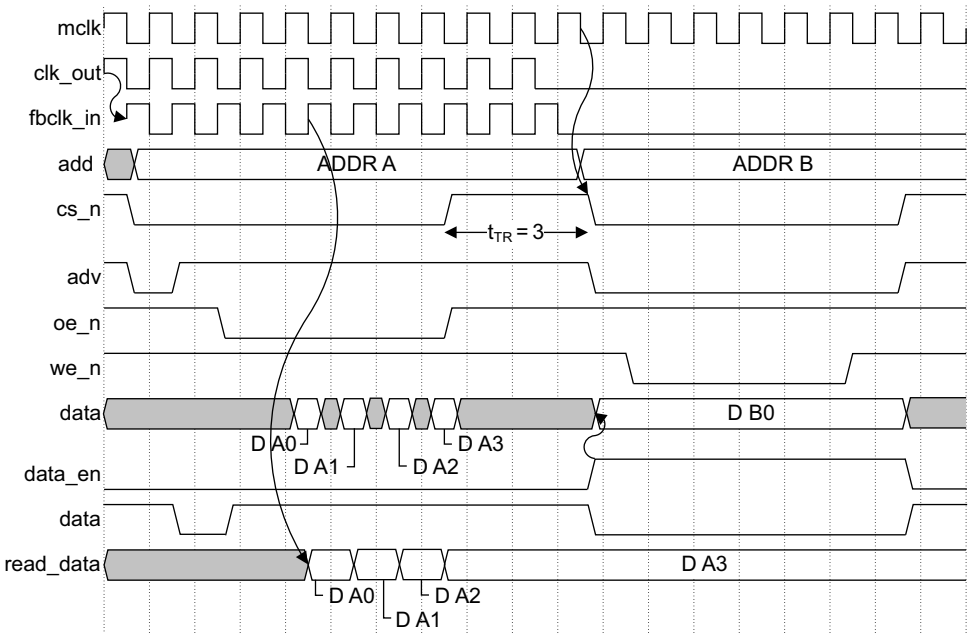


Figure 2-24 Synchronous read and asynchronous write

Programming t_{RC} and t_{WC} when the controller operates in synchronous mode

For t_{RC}:

- when using memory devices that are not wait-enabled, you must program t_{RC} to be the number of clock cycles required before valid data is available following the assertion of **cs_n**
- when using memory devices that are wait-enabled, you must program t_{RC} to be the number of clock cycles required before **wait** is active and stable, following the assertion of **cs_n**. That is:

  ```
  t_RC = 3 + t_CEOE
  ```

  Note

  t_CEOE is only required if **wait** is asserted when **oe_n** goes LOW.

For t_{WC}:

- when using memory devices that are not wait-enabled, you must program t_{WC} to be the number of clock cycles required before the first data is written, following the assertion of **cs_n**
- when using memory devices that are wait-enabled, you must program t_{WC} to be the number of clock cycles required before **wait** is active and stable, following the assertion of **cs_n**. That is:

  ```
  t_WC = 3
  ```

  Note

  If a memory device is configured so that there are two or less clock cycles between the assertion of **wait** and data being required then you must program t_{WC} as if the memory device is not wait-enabled.

Chip select assertion for SRAM memory interfaces

During repeated access to the same chip, the SMC can keep chip select asserted. To support memories that require chip select to be deasserted periodically, you can program the refresh_period_<x> Register to set a maximum number of consecutive memory bursts. You can set the number of consecutive bursts from one to 15, inclusive. See *Refresh Period 0 Register* on page 3-18 and *Refresh Period 1 Register* on page 3-19.

2.2.10 NAND interface timing diagrams

All NAND control and data outputs are registered on the rising edge of **mclkn**, which is equivalent to the falling edge of **mclk**. Additionally, read data from the memory device is registered by the SMC on the rising edge of **mclkn** before being pushed onto the read data FIFO.

Note

This section does not describe the settings for the *Operating Mode Status Register* on page 3-21 because for NAND devices you can only program the memory width field.

The following apply to NAND accesses:

Command phases

When issuing a command phase access with address cycles = 0, you must always enable at least one byte lane.

Data phases

Read data phases cannot have end commands associated with them.

Note

The internal signal **read_data** is included in the read transfer waveforms to indicate the clock edge on which data is registered by the SMC.

This section describes:

- *Command phase access*
- *Data phase access* on page 2-38
- *Command-to-data phase access timing* on page 2-39
- *Data-to-command phase access timing* on page 2-40.

Command phase access

Table 2-23 shows the settings for the *NAND Cycles Register* on page 3-21.

Table 2-23 NAND flash address input settings

Field	t_rc	t_wc	t_rea	t_wp	t_clr	t_ar	t_rr
Value	-	b0010	-	b001	-	-	-

Figure 2-25 shows an address input phase. The cycle time t_wc is set to two, and the **we_n** assertion duration, t_wp, is set to one. The address consists of three cycles, and the second command is also required.

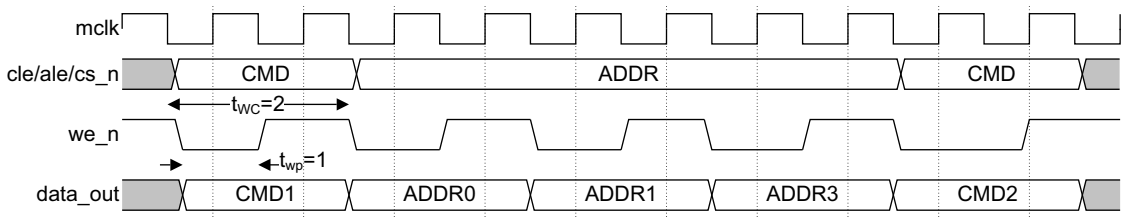


Figure 2-25 NAND flash address input

Table 2-24 shows example **awaddr** fields for NAND flash address input.

Table 2-24 NAND flash address input example awaddr fields

awaddr bits	Value	Description
[31:24]	Chip select	-
[23:21]	b011	Three address cycles
[20]	1	End command required
[19]	0	Command phase transfer
[18:11]	CMD2	-
[10:3]	CMD1	-
[2:0]	b000	Address alignment

Data phase access

Table 2-25 shows the settings for the *NAND Cycles Register* on page 3-21.

Table 2-25 NAND flash read settings

Field	t_rc	t_wc	t_rea	t_wp	t_clr	t_ar	t_rr
Value	b0011	-	b010	-	-	-	-

Figure 2-26 shows a read from NAND flash. The cycle time is set to three and the **re_n** assertion delay to two cycles. Three data items are read.

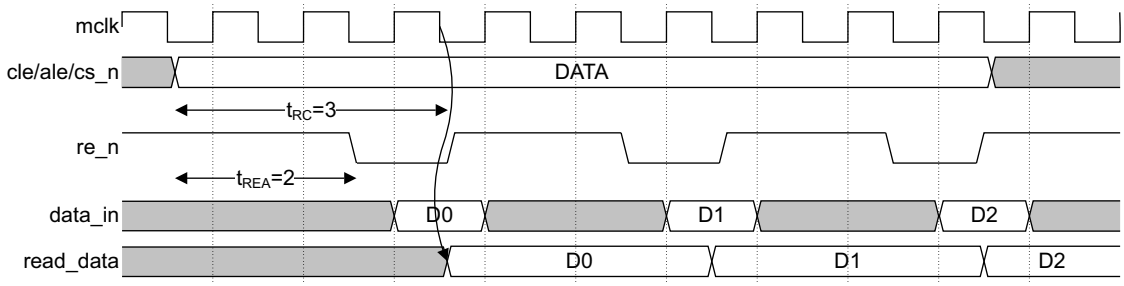


Figure 2-26 NAND flash read

Table 2-26 shows example **araddr** fields for NAND flash page read.

Table 2-26 NAND flash page read example araddr fields

araddr bits	Value	Description
[31:24]	Chip select	-
[23:22]	b00	Reserved
[21]	1	Last transfer, deassert chip select when complete
[20]	0	End command required
[19]	1	Data phase transfer
[18:11]	CMD2	-

Table 2-26 NAND flash page read example araddr fields (continued)

araddr bits	Value	Description
[10]	0	ECC Last
[9:3]	b000_0000	Reserved
[2:0]	b000	Address alignment

Command-to-data phase access timing

Table 2-27 shows the address latch to data phase settings for the *NAND Cycles Register* on page 3-21.

Table 2-27 Address latch to data phase settings

Field	t_rc	t_wc	t_rea	t_wp	t_clr	t_ar	t_rr
Value	b0011	b0010	b010	b001	-	b010	-

Figure 2-27 shows that t_{AR} is the number of extra cycles delay between address latch (**ale**) falling and the start of a new data phase command.

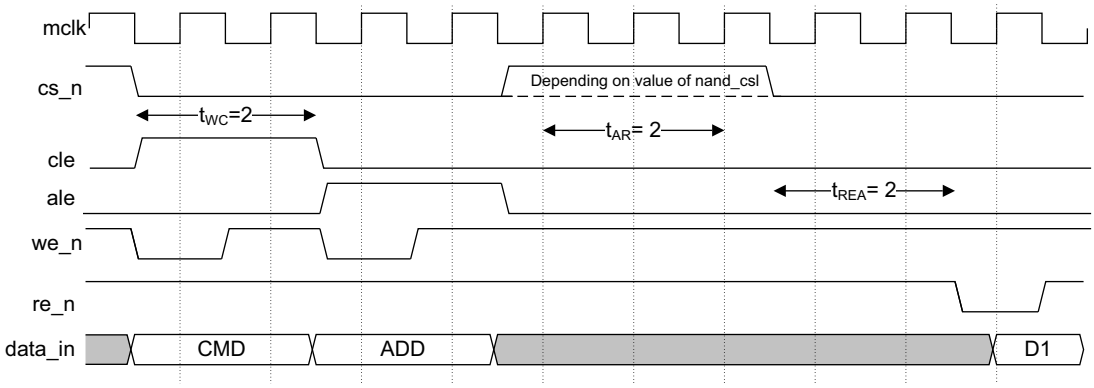


Figure 2-27 Address latch to data phase command

Table 2-28 shows the busy synchronization to data phase register settings for the *NAND Cycles Register* on page 3-21.

Table 2-28 Busy synchronization to data phase settings

Field	t_rc	t_wc	t_rea	t_wp	t_clr	t_ar	t_rr
Value	b0011	-	b010	-	-	-	b0010

When booting from NAND with **nand_booten_<x>** asserted, t_{RR} is the number of extra cycles delay between the synchronization of the **busy** signal and the start of the next data phase command as Figure 2-28 on page 2-40 shows.

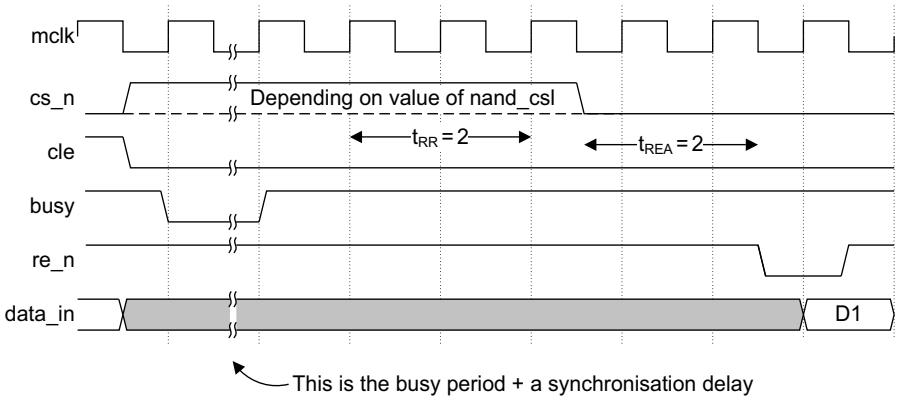


Figure 2-28 Busy synchronization to data phase command

Table 2-29 shows the command latched to data phase register settings for the *NAND Cycles Register* on page 3-21.

Table 2-29 Command latched to data phase settings

Field	t_rc	t_wc	t_rea	t_wp	t_clr	t_ar	t_rr
Value	b0011	b0010	b010	-	b010	-	-

Figure 2-29 shows the t_{CLR} delay that is the number of extra cycles delay between a command being latched, **cle** HIGH, and the start, **CS** asserted, of a data phase command.

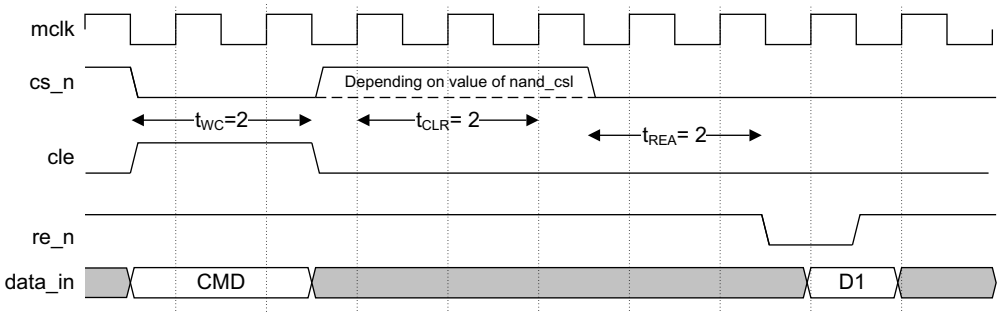


Figure 2-29 Command latched to data phase command

Note

The t_{CLR} delay is applied before both read and write data phase commands.

Data-to-command phase access timing

Table 2-30 shows the data phase to command phase register settings for the *NAND Cycles Register* on page 3-21.

Table 2-30 Data phase to command phase settings

Field	t_rc	t_wc	t_rea	t_wp	t_clr	t_ar	t_rr
Value	b0010	b0010	b001	b001	-	-	b0010

The SMC also uses t_{RR} for the number of cycles delay between the a data phase command and the assertion of the other data strobe, that is, either between:

- a write data phase and the next assertion of **re_n**
- a read data phase and the next assertion of **we_n** as Figure 2-30 shows.

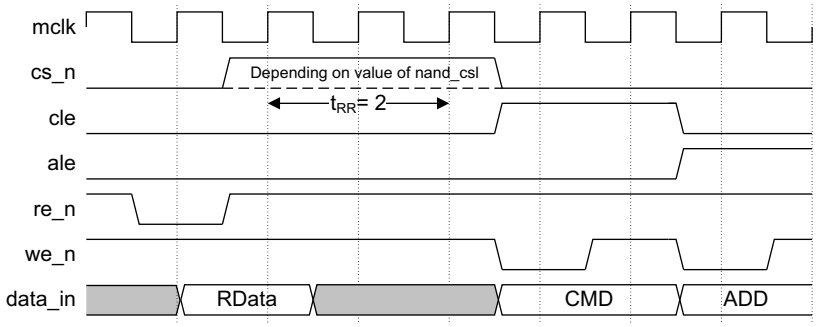


Figure 2-30 Read data phase to command phase

Note

Figure 2-28 on page 2-40 and Figure 2-30 show that the SMC uses t_{RR} in two different ways. After you determine both values for t_{RR} you must program t_{RR} with the larger of the two values.

2.2.11 Error Correction Code block

An ECC block can be included for each NAND interface at the configuration stage. It operates on a number of 512-byte frames of NAND memory and can be programmed to store the ECC codes after the data in memory. For writes, the ECC is written to the spare area of the page. For reads, the result of a frame ECC check are made available to the device driver.

Note

Because there is no standard interface for NAND memory devices, it is important to know the characteristics of a particular memory type, before you enable the SMC to use ECC functionality.

A configuration option enables an extra frame of 4, 8, 16, or 32 bytes to be included at the end of the page, before the start of the ECC data. Figure 2-31 shows the ECC block structure in memory.

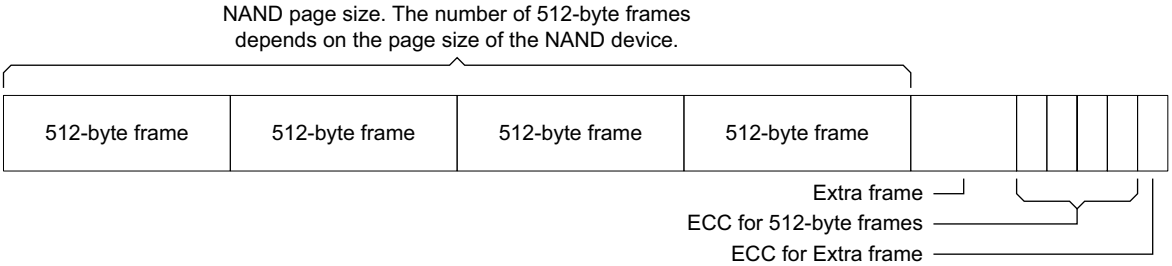


Figure 2-31 ECC block structure

The following sections describe:

- *Operation*
- *Addressing* on page 2-44
- *Data* on page 2-45
- *Address jumping* on page 2-45
- *Address modes* on page 2-46
- *Cache mode accesses* on page 2-47
- *Error codes* on page 2-48
- *Interrupts* on page 2-48
- *Correcting errors* on page 2-48.

Operation

The ECC calculation uses a simple *Hamming* code, using 1-bit correction 2-bit detection. It starts when a valid read or write command with a 512-byte aligned address is detected on the memory interface, and the block is enabled using the *ECC Configuration Register* on page 3-26. Values stored in the *ECC Command 0 Register* on page 3-28 and the *ECC Command 1 Register* on page 3-29 are used to detect the start of an address phase access.

Figure 2-32 on page 2-43 shows an overview of how the ECC operates.

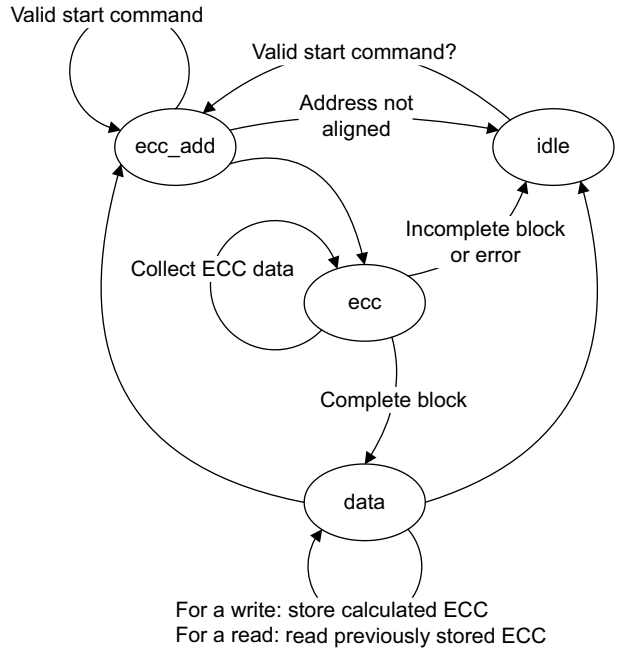


Figure 2-32 ECC state diagram

A 24-bit ECC value is generated for each 512-byte frame and a shorter code between 10 and 16 bits for the extra frame.

Note

For a 16-bit interface, ECC values are written to memory aligned to 16-bit boundaries.

Figure 2-33 on page 2-44 shows the basic operation with no reading ECC values between blocks.

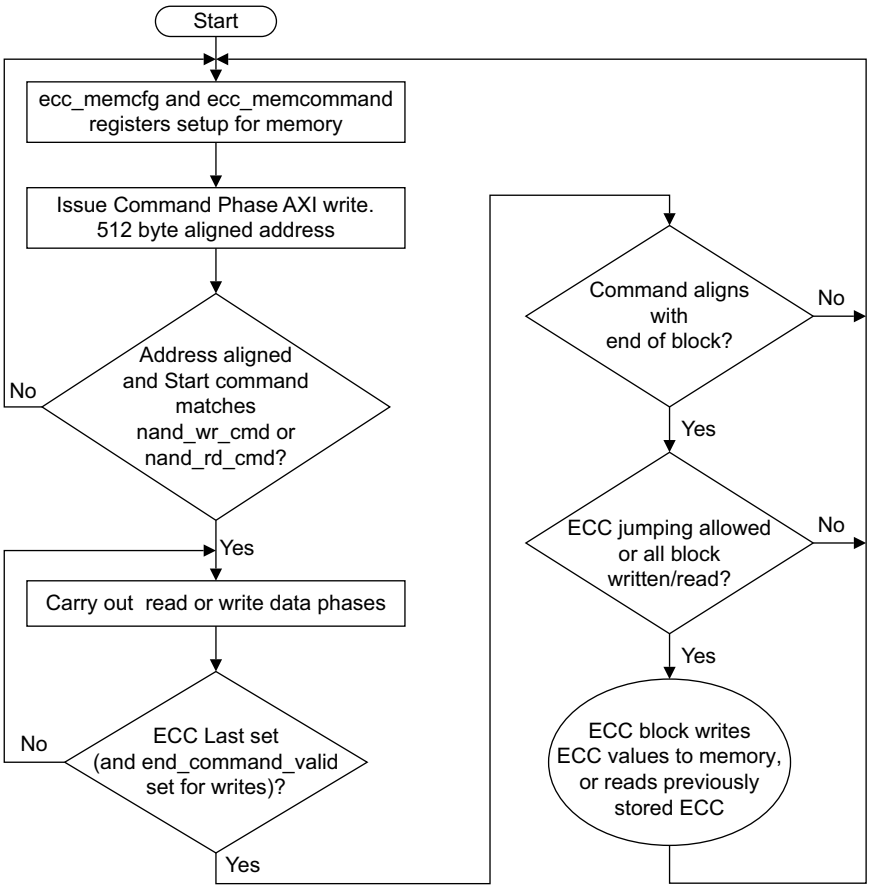


Figure 2-33 Basic operation

Addressing

The ECC block supports two addressing modes. This must be set correctly for the type of memory in use, because it is used when generating addresses to move around the NAND page, and for detecting 512-byte aligned addresses.

Normal mode addressing

The normal mode, setting the ecc_ignore_add_eight bit to 0, expects the first two bytes to contain just the column address bits as Table 2-31 shows.

Table 2-31 Normal mode addressing

Cycle	I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
1st	A7	A6	A5	A4	A3	A2	A1	A0
2nd	L[a]	L[a]	L[a]	L[a]	A11[b]/L	A10	A9	A8
3rd-7th	Don't care							

a. These bits must be LOW otherwise the behavior is undefined.

b. A11 might be present, depending on the memory width.

This mode supports all random access, column change commands, and up to four 512-byte frames. See *Address jumping* on page 2-45.

Secondary mode addressing

The secondary mode, setting the ecc_ignore_add_eight bit to 1, supports memories with 512 bits where the address formatting is as Table 2-32 shows.

Table 2-32 Secondary mode addressing

Cycle	I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
1st	A7	A6	A5	A4	A3	A2	A1	A0
2nd	Don't care							
3rd-7th	Don't care							

Note

In this mode, A8 is passed as part of the start command and is not present in the data.

In this mode, random accesses are not possible. The nand_rd_col_change field in the *ECC Command 1 Register* on page 3-29 can be used as an alternative read start command. This enables ECC calculation on just the extra frame at the end of the page.

For writes, issuing a zero address cycle, pointer change command, that matches the `nand_rd_col_change` command, tells the ECC block that the next write command is to the extra bits. This only applies to the subsequent write command, even if the memory only requires one pointer access for multiple writes.

Data

When a valid start address has been sent, data can be read or written using a series of NAND data phase commands. See *NAND data phase transfers* on page 2-16. The last access must align with the end of a 512-byte frame, or the extra frame if it is enabled. You must set ECC Last on the last data phase access, to tell the ECC block not to expect any more data.

If an access to a different chip is received during an ECC operation, the ECC block aborts and sets the ecc_last_status field in the *ECC Status Register* on page 3-24 to indicate *Data stop after incomplete block*. No more ECC data is read or written to memory.

Address jumping

To enable you to write individual 512-byte frames, or the ECC extra frame, the SMC can issue address phase commands to move around the NAND page.

The ecc_jump field in the *ECC Configuration Register* on page 3-26 controls how the SMC jumps to the correct place in memory. You can program ecc_jump to:

Jump using full command

The SMC uses the commands stored in the ecc_memcommand1 Register to control the NAND address pointer. It also uses these commands to detect the start of a NAND read or write. See *ECC Command 0 Register* on page 3-28.

Jump using column change commands

The SMC uses the commands stored in the ecc_memcommand2 Register. The value used as the end command for a write access is taken directly from the previous AXI command that had the end_command_valid bit asserted. See *ECC Command 1 Register* on page 3-29.

No jumping

The SMC only reads or writes ECC data at the end of a page.

The ECC values for writes are only written to memory after the end command is received. For reads, the ecc_read_end bit setting can be used to read ECC data from memory between every frame.

Address modes

The following sections describe the different methods used to control the address pointer, when writing ECC values:

- *ecc_jump = no jumping*
- *ecc_jump = column change*
- *ecc_jump = full command* on page 2-47
- *ignore_add_8 and ecc_jump is not no_jump* on page 2-47.

Note

The same methods can be applied to reads, except that end commands may be output after the address, if enabled in the ecc_memcommand<x> Registers, but never after a data transfer.

ecc_jump = no jumping

If the ecc_jump field is set to no jumping, and not all frames in a page are read or written, then an error is generated. However, the calculated ECC values are available in the ecc_value<x> Registers. If required, you can then use software to write them to memory. See Figure 2-34.

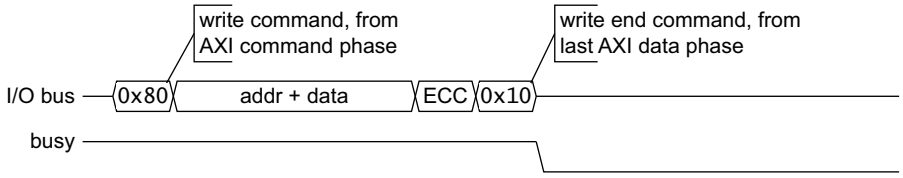


Figure 2-34 Every block is written

Note

The command values shown in these diagrams, for example `0x80`, `0x10`, or `0x15`, are representative and may not match your particular NAND device.

ecc_jump = column change

If the ecc_jump field is set to column change commands, the SMC issues a `col_change` command, with two address cycles. See Figure 2-35.

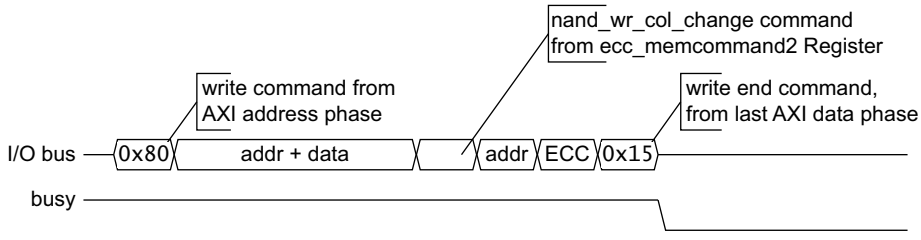


Figure 2-35 Not every block written, random access

ecc_jump = full command

If a full command is used, the SMC issues an entire new command phase access with the same number of address cycles as the initial write. See Figure 2-36.

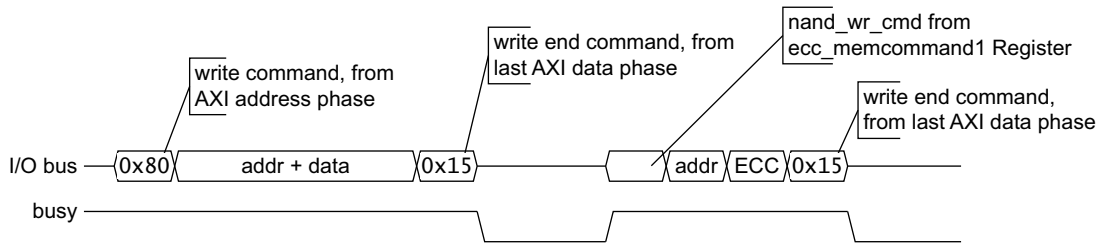


Figure 2-36 Full jumping

Note

- If the ecc_jump field is set to use full commands, this counts against the maximum number of program operations before a NAND page must be erased.
- If the ecc_read_end bit is set to read between frames, then each boundary must be aligned with the end of a data phase access. Otherwise, data phases accesses can cross boundaries between frames.

ignore_add_8 and ecc_jump is not no_jump

If not all frames are written, the SMC issues a pointer change command using the value in the nand_rd_col_change field of the *ECC Command 1 Register* on page 3-29. For reads, the nand_rd_col_change field is used instead of the standard read command, to access the extra bits at the end of the page. See Figure 2-37.

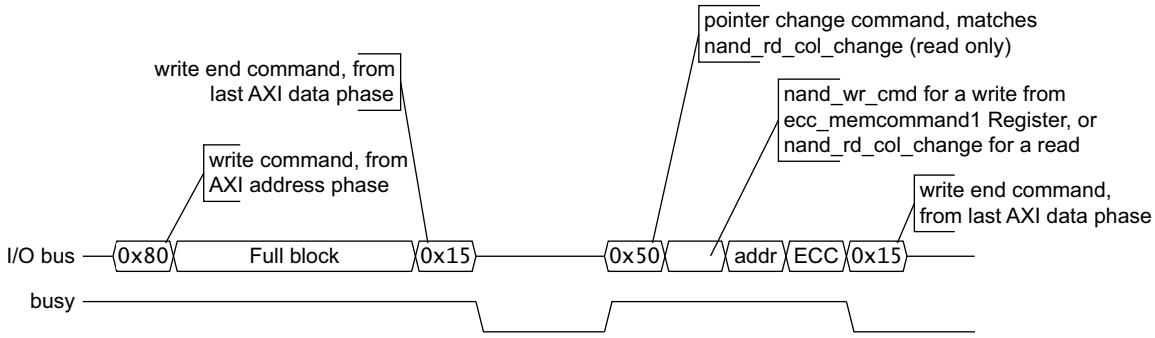


Figure 2-37 addressing mode ignore_add_8 with extra blocks

Note

After writing or reading ECC values in the secondary addressing mode, see *Secondary mode addressing* on page 2-45, the ECC block does not return the pointer to its previous state. Software might have to correct the pointer, depending on the memory and if the ECC block was forced to jump into the extra data area.

Cache mode accesses

If performing cache mode reads, the entire page must be read and ECC Last only issued on the last data phase access of the last page. Undefined behavior occurs if you attempt to read data beyond the page size.

Note

- The ecc_jump field must be set to no jump to prevent the SMC from attempting to move the address pointer around the cache register.
- If multiple pages are read, then the software must maintain a count of the number of pages. All block valid and read flags are cleared when the first frame of a new page is read.

Error codes

The error code available from the *ECC Status Register* on page 3-24 applies to the previous ECC operation. It must only be considered valid when the ECC block is not busy.

Interrupts

Interrupts can be generated:

- when the ECC block detects an error on a read
- when the ECC data is read from memory, if the ecc_int_pass bit is set in the *ECC Configuration Register* on page 3-26
- when an error occurs, if the ecc_int_abort bit is set in the *ECC Configuration Register* on page 3-26.

Interrupts can be cleared by:

- writing to the interrupt flag in the *ECC Status Register* on page 3-24
- writing any value to the appropriate *ECC Block Registers* on page 3-30.

Note

To enable the external interrupt, the ecc_int_enable0 or ecc_int_enable1 bits must be set using the *Set Configuration Register* on page 3-11.

Correcting errors

The SMC identifies the occurrence and location of errors so that software can correct those errors.

If an error occurs, the ecc_fail bit for that frame is set in the *ECC Status Register* on page 3-24. If the error is correctable, then the ecc_correct bit is set in the corresponding ecc_value<x> Register and the ecc_value field provides the location of the bit that must be corrected. See *ECC Block Registers* on page 3-30.

Table 2-33 shows the decoded meaning of the ecc_fail bit and the ecc_correct bit.

Table 2-33 ecc_fail bit and ecc_can_correct bit encoding

ecc_fail bit	ecc_can_correct bit	Meaning
0	0	No error
0	1	Parity error
1	0	Multiple error
1	1	Single error

The bottom three bits in the ecc_value field provide the bit number, and the remaining 21 bits indicate which byte contains an error. For example, an ecc_value of `0x101` indicates that bit 1 of byte 32 is incorrect.

Chapter 3
Programmers Model

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

- *About the programmers model* on page 3-2
- *Register summary* on page 3-5
- *Register descriptions* on page 3-8.

Note

See also Chapter 6 *Configurations*.

3.1 About the programmers model

The following information applies to the SMC registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these location can result in Unpredictable behavior.
- Unless otherwise stated in the accompanying text:
  - do not modify undefined register bits
  - ignore undefined register bits on reads
  - all register bits are reset to a logic 0 by a system or power-on reset.
- Access type in *Register summary* on page 3-5 is described as follows:
  RW Read and write.
  RO Read only.
  WO Write only.

3.1.1 Register map

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

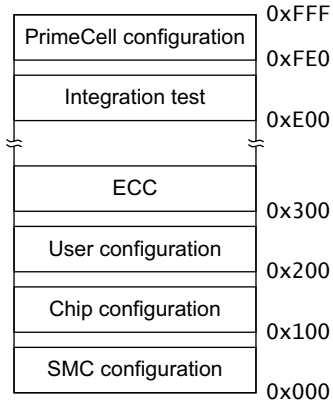


Figure 3-1 Register map

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

- *Memory controller configuration registers*
- *Chip select configuration registers* on page 3-3
- *User configuration registers* on page 3-3
- *ECC registers* on page 3-3
- *Integration test registers* on page 3-4
- *CoreLink ID registers* on page 3-4.

Memory controller configuration registers

Use these registers for the global configuration, and control of the operating state, of the SMC.

Chip select configuration registers

These registers hold the operating parameters of each chip select. If the SMC is not configured to support all chip selects, the corresponding registers are not implemented.

Figure 3-2 shows the chip<*n*> configuration register map, where <*n*> = 0 to 3.

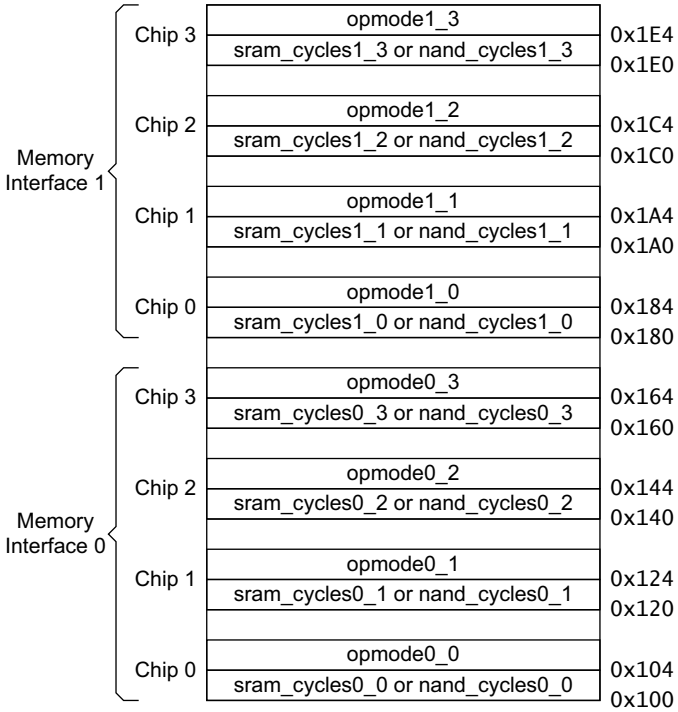


Figure 3-2 Chip<n> configuration register map

Note

Figure 3-2 shows the maximum number of supported chips. If you use less chips then the unused chip configuration blocks are read back as zero.

User configuration registers

These registers provide general purpose I/O for user-specific applications.

Figure 3-3 shows the user configuration memory register map.

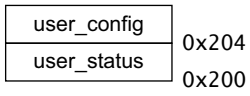


Figure 3-3 User configuration register map

ECC registers

Figure 3-4 on page 3-4 shows the ECC register map.

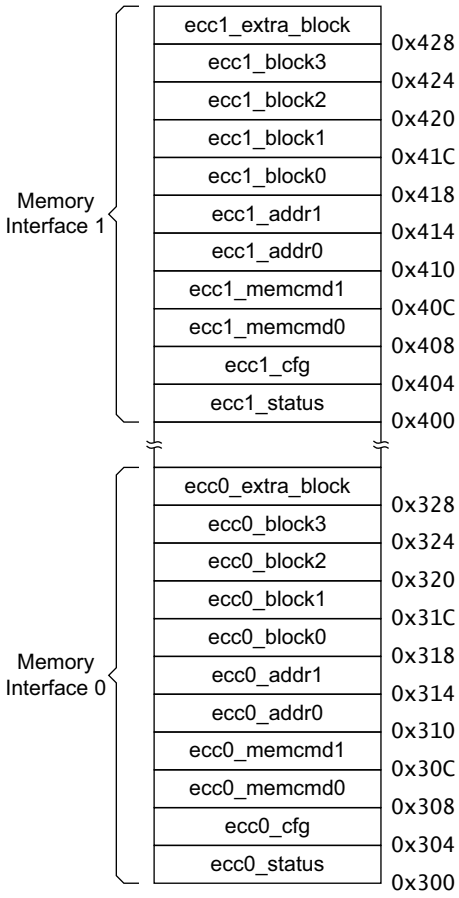


Figure 3-4 ECC register map

Integration test registers

Use these registers to verify correct integration of the SMC within a system, by enabling non-AMBA signals to be set and read.

CoreLink ID registers

These registers enable the identification of system components by software. Figure 3-5 shows the CoreLink configuration register map.

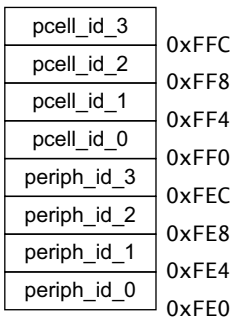


Figure 3-5 CoreLink configuration register map

3.2 Register summary

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

Table 3-1 Register summary

Offset	Name	Type	Reset	Description
0x000	memc_status	RO	0x00000000	*Memory Controller Status Register* on page 3-8.
0x004	memif_cfg	RO	Configuration dependent	*Memory Interface Configuration Register* on page 3-9.
0x008	mem_cfg_set	WO	-	*Set Configuration Register* on page 3-11.
0x00C	mem_cfg_clr	WO	-	*Clear Configuration Register* on page 3-12.
0x010	direct_cmd	WO	-	*Direct Command Register* on page 3-13.
0x014	set_cycles	WO	-	*Set Cycles Register* on page 3-15.
0x018	set_opmode	WO	-	*Set Operating Mode Register* on page 3-16.
0x020	refresh_0	RW	0x00000000	*Refresh Period 0 Register* on page 3-18.
0x024	refresh_1	RW	0x00000000	*Refresh Period 1 Register* on page 3-19.
0x028-0x0FC	-	-	-	Reserved, read undefined, write as zero.
0x100	sram_cycles0_0 or nand_cycles0_0	RO	0x0002B3CC for sram_cycles Registers. 0x0024ABCC for nand_cycles Registers.	*SRAM Cycles Register* on page 3-20. *NAND Cycles Register* on page 3-21.
0x120	sram_cycles0_1 or nand_cycles0_1			
0x140	sram_cycles0_2 or nand_cycles0_2			
0x160	sram_cycles0_3 or nand_cycles0_3			
0x180	sram_cycles1_0 or nand_cycles1_0[b]			
0x1A0	sram_cycles1_1 or nand_cycles1_1[b]			
0x1C0	sram_cycles1_2 or nand_cycles1_2[b]			
0x1E0	sram_cycles1_3 or nand_cycles1_3[b]			

Table 3-1 Register summary (continued)

Offset	Name	Type	Reset	Description
0x104	opmode0_0	RO	Configuration dependent[a]	*Operating Mode Status Register* on page 3-21.
0x124	opmode0_1			
0x144	opmode0_2			
0x164	opmode0_3			
0x184	opmode1_0[b]			
0x1A4	opmode1_1[b]			
0x1C4	opmode1_2[b]			
0x1E4	opmode1_3[b]			
0x200	user_status	RO	Application dependent	*User Status Register* on page 3-23.
0x204	user_config	WO	-	*User Config Register* on page 3-24.
0x208-0x2FC	-	-	-	Reserved, read undefined, write as zero.
0x300 0x400	ecc0_status ecc1_status[b]	RW	0x00000000	*ECC Status Register* on page 3-24.
0x304 0x404	ecc0_cfg ecc1_cfg[b]	RW	0x00000043	*ECC Configuration Register* on page 3-26.
0x308 0x408	ecc0_memcmd0 ecc1_memcmd0[b]	RW	0x01300080	*ECC Command 0 Register* on page 3-28.
0x30C 0x40C	ecc0_memcmd1 ecc1_memcmd1[b]	RW	0x01E00585	*ECC Command 1 Register* on page 3-29.
0x310 0x410	ecc0_addr0 ecc1_addr0[b]	RO	0x00000000	*ECC Address 0 Register* on page 3-29.
0x314 0x414	ecc0_addr1 ecc1_addr1[b]	RO	0x00000000	*ECC Address 1 Register* on page 3-30.
0x318	ecc0_block0	RW	0x00000000	*ECC Block Registers* on page 3-30.
0x31C	ecc0_block1			
0x320	ecc0_block2			
0x324	ecc0_block3			
0x418	ecc1_block0[b]			
0x41C	ecc1_block1[b]			
0x420	ecc1_block2[b]			
0x424	ecc1_block3[b]			
0x328 0x428	ecc0_extra_block ecc1_extra_block[b]	RW	0x00000000	*ECC Extra Block Register* on page 3-31.

Table 3-1 Register summary (continued)

Offset	Name	Type	Reset	Description
0x32C-0x3FC 0x42C-0x4FC	-	-	-	Reserved, read undefined, write as zero.
0xE00 0xE04 0xE08	int_cfg int_inputs int_outputs	See Chapter 4 *Programmers Model for Test* for more information about these registers.		
0xE0C-0xFDC	-	-	-	Reserved, read undefined, write as zero.
0xFE0-0xFEC	periph_id_n	RO	0x00_4135_[c]	*Peripheral Identification Registers 0-3* on page 3-32.
0xFF0-0xFFC	pcell_id_n	RO	0xB105F00D	*CoreLink Identification Registers 0-3* on page 3-34.

a. Bits[1:0] and [31:16] are dependent on external tie-offs. The remaining bits default to 0.

b. Available for SMC variants that provide two memory interfaces, that is, SMC-353 and SMC-354.

c. Dependent on the variant of the SMC and the revision of the SMC. See *Peripheral Identification Register 0* on page 3-33 and *Peripheral Identification Register 2* on page 3-34

3.3 Register descriptions

This section describes the SMC registers.

3.3.1 Memory Controller Status Register

The memc_status Register characteristics are:

Purpose Provides information about the configuration and current state of the SMC.

Usage constraints Not accessible in the Reset state.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-6 shows the memc_status Register bit assignments.

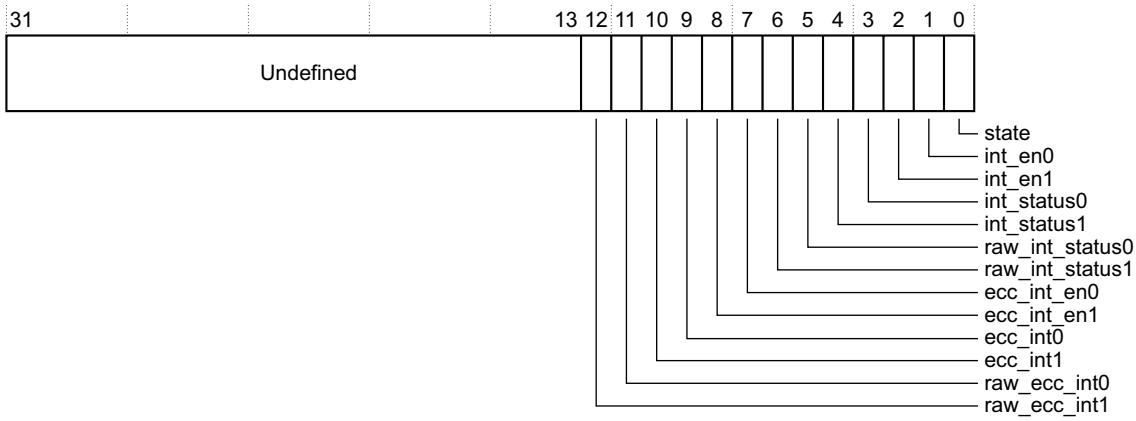


Figure 3-6 memc_status Register bit assignments

Table 3-2 shows the memc_status Register bit assignments.

Table 3-2 memc_status Register bit assignments

Bits	Name	Function
[31:13]	-	Reserved, read undefined
[12]	raw_ecc_int1	Raw status of the **ecc_int1** interrupt signal.
[11]	raw_ecc_int0	Raw status of the **ecc_int0** interrupt signal.
[10	ecc_int1	Status of the **ecc_int1** interrupt signal after ANDing with its enable bit, ecc_int_en1.
[9]	ecc_int0	Status of the **ecc_int0** interrupt signal after ANDing with its enable bit, ecc_int_en0.
[8]	ecc_int_en1	Interrupt enable status for **ecc_int1**: 0 = Interrupt is disabled so **ecc_int1** is LOW. 1 = Interrupt is enabled.
[7]	ecc_int_en0	Interrupt enable status for **ecc_int0**: 0 = Interrupt is disabled so **ecc_int0** is LOW. 1 = Interrupt is enabled.
[6]	raw_int_status1	Raw status of the **smc_int1** interrupt signal.

Table 3-2 memc_status Register bit assignments (continued)

Bits	Name	Function
[5]	raw_int_status0	Raw status of the **smc_int0** interrupt signal.
[4]	int_status1	Status of the **smc_int1** interrupt signal after ANDing with its enable bit, int_en1.
[3]	int_status0	Status of the **smc_int0** interrupt signal after ANDing with its enable bit, int_en0.
[2]	int_en1	Interrupt enable status for **smc_int1**: 0 = Interrupt is disabled so **smc_int1** is LOW. 1 = Interrupt is enabled.
[1]	int_en0	Interrupt enable status for **smc_int0**: 0 = Interrupt is disabled so **smc_int0** is LOW. 1 = Interrupt is enabled.
[0]	state	Operating state of the SMC: 0 = SMC is in the Ready state 1 = SMC is in the Low-power state.

3.3.2 Memory Interface Configuration Register

The memif_cfg Register characteristics are:

Purpose Provides information about the configuration of the memory interface.

Usage constraints Not accessible in the Reset state.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-7 shows the memif_cfg Register bit assignments.

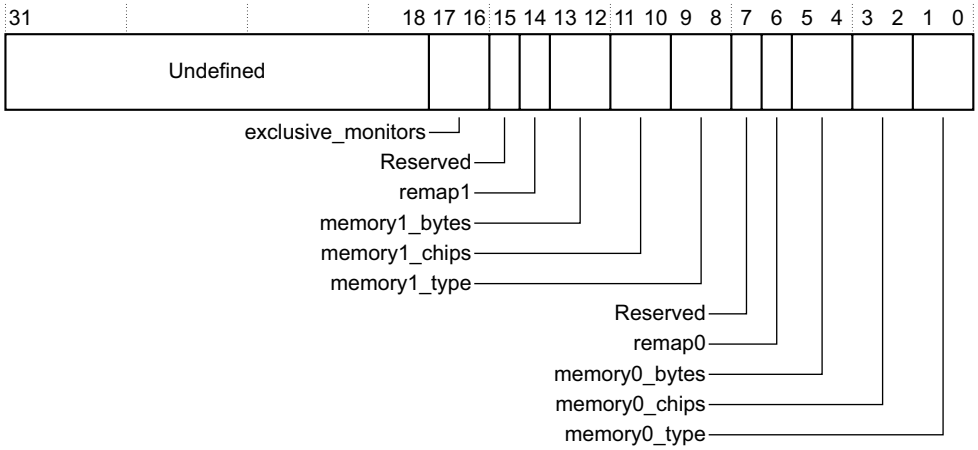


Figure 3-7 memif_cfg Register bit assignments

Table 3-3 shows the memif_cfg Register bit assignments.

Table 3-3 memif_cfg Register bit assignments

Bits	Name	Function
[31:18]	-	Reserved, read undefined.
[17:16]	exclusive_monitors	Returns the number of exclusive access monitor resources that are implemented in the SMC. b00 = 0 monitors b01 = 1 monitor b10 = 2 monitors b11 = 4 monitors. See *Exclusive accesses* on page 2-12.
[15]	-	Reserved, read undefined.
[14]	remap1	Returns the value of the **remap_1** input. See *Miscellaneous signals* on page B-3.
[13:12]	memory1_bytes	Returns the maximum width of the SMC memory data bus for interface 1: b00 = 8 bits b01 = 16 bits b10 = 32 bits b11 = reserved.
[11:10]	memory1_chips	Returns the number of different chip selects that the memory interface 1 supports: b00 = 1 chip b01 = 2 chips b10 = 3 chips b11 = 4 chips.
[9:8]	memory1_type	Returns the memory interface 1 type: b00 = Configuration does not include this memory interface b01 = SRAM non-multiplexed b10 = NAND b11 = SRAM multiplexed. If b00, the remaining bit slices for memory interface 1 are always read as 0.
[7]	-	Reserved, read undefined.
[6]	remap0	Returns the value of the **remap_0** input. See *Miscellaneous signals* on page B-3.

Table 3-3 memif_cfg Register bit assignments (continued)

Bits	Name	Function
[5:4]	memory0_bytes	Returns the maximum width of the SMC memory data bus for interface 0: b00 = 8 bits b01 = 16 bits b10 = 32 bits b11 = reserved.
[3:2]	memory0_chips	Returns the number of different chip selects that the memory interface 0 supports: b00 = 1 chip b01 = 2 chips b10 = 3 chips b11 = 4 chips.
[1:0]	memory0_type	Returns the memory interface 0 type: b00 = reserved b01 = SRAM non-multiplexed b10 = NAND b11 = SRAM multiplexed.

3.3.3 Set Configuration Register

The mem_cfg_set Register characteristics are:

Purpose Use this to:
- enable interrupts
- request the SMC to enter the Low-power state.

Usage constraints Not accessible in the Reset state.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-8 shows the mem_cfg_set Register bit assignments.

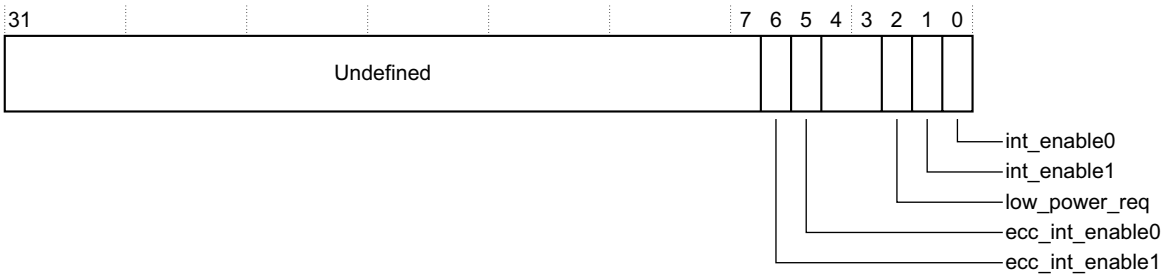


Figure 3-8 mem_cfg_set Register bit assignments

Table 3-4 shows the mem_cfg_set Register bit assignments.

Table 3-4 mem_cfg_set Register bit assignments

Bits	Name	Function
[31:7]	-	Reserved, write as zero.
[6]	ecc_int_enable1	0 = No effect 1 = Enables the **ecc_int1** interrupt. SMC sets the ecc_int_en1 bit to 1 in the memc_status Register, see *Memory Controller Status Register* on page 3-8.
[5]	ecc_int_enable0	0 = No effect 1 = Enables the **ecc_int0** interrupt. SMC sets the ecc_int_en0 bit to 1 in the memc_status Register, see *Memory Controller Status Register* on page 3-8.
[4:3]	-	Reserved, write as zero.
[2]	low_power_req	0 = No effect 1 = Requests the SMC to enter Low-power state when it next becomes idle.
[1]	int_enable1	0 = No effect 1 = Enables the **smc_int1** interrupt. SMC sets the int_en1 bit to 1 in the memc_status Register, see *Memory Controller Status Register* on page 3-8.
[0]	int_enable0	0 = No effect 1 = Enables the **smc_int0** interrupt. SMC sets the int_en0 bit to 1 in the memc_status Register, see *Memory Controller Status Register* on page 3-8.

3.3.4 Clear Configuration Register

The mem_cfg_clr Register characteristics are:

Purpose Use this to:
- disable interrupts
- request the SMC to exit the Low-power state.

Usage constraints Not accessible in the Reset state.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-9 shows the mem_cfg_clr Register bit assignments.

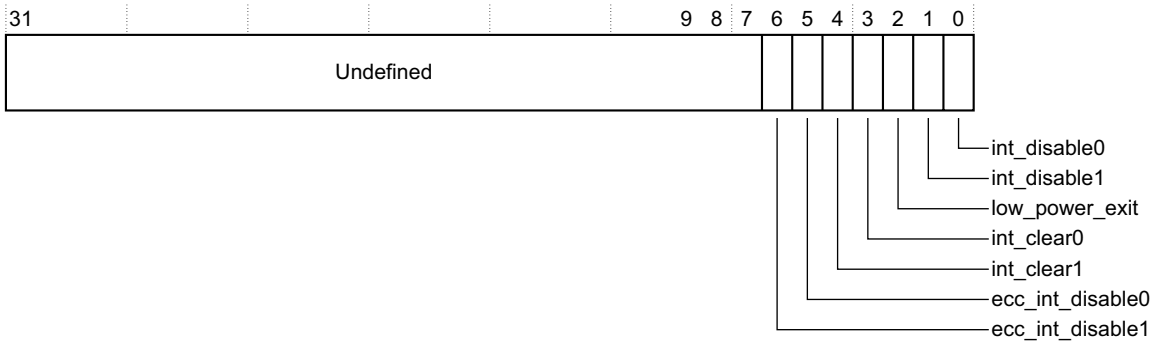


Figure 3-9 mem_cfg_clr Register bit assignments

Table 3-5 shows the mem_cfg_clr Register bit assignments.

Table 3-5 mem_cfg_clr Register bit assignments

Bits	Name	Function
[31:7]	-	Reserved, write as zero.
[6]	ecc_int_disable1	0 = No effect 1 = Disables the **ecc_int1** interrupt. SMC sets the ecc_int_en1 bit to 0 in the memc_status Register, see *Memory Controller Status Register* on page 3-8
[5]	ecc_int_disable0	0 = No effect 1 = Disables the **ecc_int0** interrupt. SMC sets the ecc_int_en0 bit to 0 in the memc_status Register, see *Memory Controller Status Register* on page 3-8
[4]	int_clear1	0 = No effect 1 = Clear SMC Interrupt 1, as an alternative to an AXI read.
[3]	int_clear0	0 = No effect 1 = Clear SMC Interrupt 0, as an alternative to an AXI read.
[2]	low_power_exit	0 = No effect 1 = Request the SMC to exit Low-power state.
[1]	int_disable1	0 = No effect 1 = Disables the **smc_int1** interrupt. SMC sets the int_en1 bit to 0 in the memc_status Register, see *Memory Controller Status Register* on page 3-8
[0]	int_disable0	0 = No effect 1 = Disables the **smc_int0** interrupt. SMC sets the int_en0 bit to 0 in the memc_status Register, see *Memory Controller Status Register* on page 3-8

3.3.5 Direct Command Register

The direct_cmd Register characteristics are:

Purpose Initializes and updates the external memory devices using the data in the:
- *Set Cycles Register* on page 3-15
- *Set Operating Mode Register* on page 3-16.

Usage constraints You cannot write to this register in the Reset or Low-power states.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-10 shows the direct_cmd Register bit assignments.

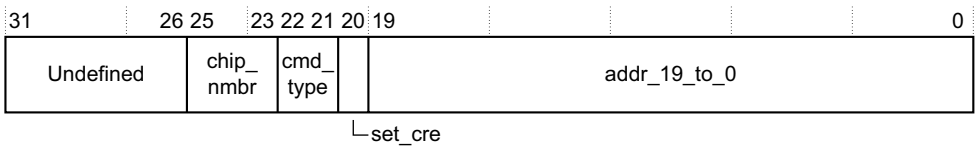


Figure 3-10 direct_cmd Register bit assignments

Table 3-6 shows the direct_cmd Register bit assignments.

Table 3-6 direct_cmd Register bit assignments

Bits	Name	Function
[31:26]	-	Reserved, write as zero.
[25:23]	chip_nmbr	Selects chip configuration register bank to update, and enables chip mode register access depending on cmd_type. The encoding is: b000-b011 = Chip selects 1-4 on memory interface 0 b100-b111 = Chip selects 1-4 on memory interface 1.
[22:21]	cmd_type	Selects the command type: b00 = UpdateRegs and AXI b01 = ModeReg b10 = UpdateRegs b11 = ModeReg and UpdateRegs.
[20]	set_cre	Maps to the configuration register enable signal, **cre**, when a ModeReg command is issued. The encoding is: 0 = **cre** is LOW 1 = **cre** is HIGH when ModeReg write occurs.
[19:0]	addr_19_to_0	When cmd_type = UpdateRegs and AXI then: • bits [15:0] are used to match **wdata[15:0]** • bits [19:16] are reserved. Write as zero. When cmd_type = ModeReg or ModeReg and UpdateRegs, these bits map to the external memory address bits [19:0]. When cmd_type = UpdateRegs, these bits are reserved. Write as zero.

In Table 3-6 the cmd_type assignments are:

ModeReg Programs the configuration registers in a memory device.

UpdateRegs The SMC copies the contents of the:

- *Set Operating Mode Register* on page 3-16 to the *Operating Mode Status Register* on page 3-21
- *Set Cycles Register* on page 3-15 to the *SRAM Cycles Register* on page 3-20 for SRAM devices or *NAND Cycles Register* on page 3-21 for NAND devices.

Note

The SMC uses the chip_nmbr field to select the appropriate destination register.

ModeReg and UpdateRegs

The SMC performs, at the same time, the operations that the ModeReg cmd_type and the UpdateRegs cmd_type specify. This combined command enables you to modify memory configurations, while an access to memory is in progress. This enables code to be executed from memory while simultaneously, from the software perspective, moving the memory device to a different operating mode. The SMC achieves this by synchronizing the update of the chip configuration register with the memory configuration register write.

UpdateRegs and AXI

Use this to synchronize register updates when the memory is configured using a sequence of AXI commands. The addr_19_to_0 field in the direct_cmd Register is compared against the AXI write data to control when the SMC updates the following registers:

- *Operating Mode Status Register* on page 3-21
- *SRAM Cycles Register* on page 3-20 for SRAM devices or *NAND Cycles Register* on page 3-21 for NAND devices.

Note

The SMC uses the chip_nmbr field to select the appropriate destination register.

This method is required for NOR Flash devices.

3.3.6 Set Cycles Register

The set_cycles Register characteristics are:

Purpose Contains configuration data for the external memory devices. The data is written to a memory device when the SMC receives a write to the *Direct Command Register* on page 3-13. See *Memory manager operation* on page 2-22 for more information.

Usage constraints You cannot write to this register in the Reset or Low-power states.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-11 shows the set_cycles Register bit assignments.

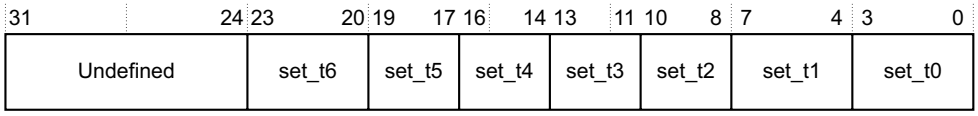


Figure 3-11 set_cycles Register bit assignments

Table 3-7 shows the set_cycles Register bit assignments.

Table 3-7 set_cycles Register bit assignments

Bits	Name	Function
[31:24]	-	Reserved, write as zero.
[23:20]	set_t6	Contains the value to be written to either the: • we_time bit of the *SRAM Cycles Register* on page 3-20[a] • t_rr field of the *NAND Cycles Register* on page 3-21.
[19:17]	set_t5	Contains the value to be written to either the: • t_tr field of the *SRAM Cycles Register* on page 3-20 • t_ar field of the *NAND Cycles Register* on page 3-21.
[16:14]	set_t4	Contains the value to be written to either the: • t_pc field of the *SRAM Cycles Register* on page 3-20 • t_clr field of the *NAND Cycles Register* on page 3-21.

Table 3-7 set_cycles Register bit assignments (continued)

Bits	Name	Function
[13:11]	set_t3	Contains the value to be written to the t_wp field in either the: • *SRAM Cycles Register* on page 3-20[b] • *NAND Cycles Register* on page 3-21.
[10:8]	set_t2	Contains the value to be written to either the: • t_ceoe field of the *SRAM Cycles Register* on page 3-20[c] • t_rea field of the *NAND Cycles Register* on page 3-21.
[7:4]	set_t1	Contains the value to be written to the t_wc field in either the: • *SRAM Cycles Register* on page 3-20 • *NAND Cycles Register* on page 3-21.
[3:0]	set_t0	Contains the value to be written to the t_rc field in either the: • *SRAM Cycles Register* on page 3-20 • *NAND Cycles Register* on page 3-21.

a. Permitted values are `0x0` and `0x1`.

b. Permitted value is $t\_wp_{MAX} \leq t\_wc - 1$.

c. Permitted value is $t\_ceoe_{MAX} \leq t\_rc - 1$.

3.3.7 Set Operating Mode Register

The set_opmode Register characteristics are:

Purpose This write-only register is the holding register for the opmode<x>_<n> Registers. It contains configuration data for the external memory devices. The data is written to a memory device when the SMC receives a write to the *Direct Command Register* on page 3-13. See *Memory manager operation* on page 2-22 for more information.

Usage constraints You cannot write to this register in the Reset or Low-power states.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-12 shows the set_opmode Register bit assignments.

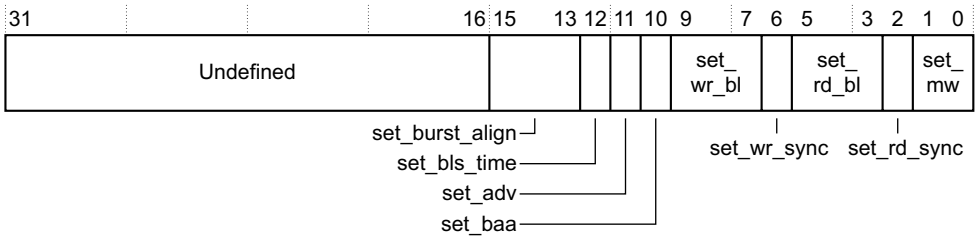


Figure 3-12 set_opmode Register bit assignments

Table 3-8 shows the set_opmode Register bit assignments.

Table 3-8 set_opmode Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved, write as zero.
[15:13]	set_burst_align	Contains the value to be written to the specific SRAM chip opmode Register burst_align field. When you configure the SMC to perform synchronous transfers[a], these bits control if memory bursts are split on memory burst boundaries: b000 = bursts can cross any address boundary b001 = burst split on memory burst boundary, that is, 32 beats for continuous b010 = burst split on 64 beat boundary b011 = burst split on 128 beat boundary b100 = burst split on 256 beat boundary b101-b111 = reserved. For a NAND memory interface these bits are reserved, and written as zero.
[12]	set_bls_time	Contains the value to be written to the specific SRAM chip opmode Register *byte lane strobe* (bls) bit. This bit affects the assertion of the byte lane strobe outputs. 0 = bls timing equals chip select timing. This is the default setting. 1 = bls timing equals **we_n** timing. This setting is used for eight memories that have no **bls_n** inputs. In this case, the **bls_n** output of the SMC is connected to the **we_n** memory input. For a NAND memory interface this bit is reserved, and written as zero.
[11]	set_adv	Contains the value to be written to the specific SRAM chip opmode Register *address valid (*adv) bit. The memory uses the address advance signal **adv_n** when set. For a NAND memory interface this bit is reserved, and written as zero.
[10]	set_baa	Contains the value to be written to the specific SRAM chip opmode Register *burst address advance* (baa) bit. The memory uses the **baa_n** signal when set. For a NAND memory interface this bit is reserved, and written as zero.
[9:7]	set_wr_bl	Contains the value to be written to the specific SRAM chip opmode Register wr_bl field. Encodes the memory burst length: b000 = 1 beat b001 = 4 beats b010 = 8 beats b011 = 16 beats b100 = 32 beats b101 = continuous b110-b111 = reserved. For a NAND memory interface these bits are reserved, and written as zero.
[6]	set_wr_sync	Contains the value to be written to the specific SRAM chip opmode Register wr_sync bit. The memory writes are synchronous when set. For a NAND memory interface this bit is reserved, and written as zero.

Table 3-8 set_opmode Register bit assignments (continued)

Bits	Name	Function
[5:3]	set_rd_bl	Contains the value to be written to the specific SRAM chip opmode Register rd_bl field. Encodes the memory burst length: b000 = 1 beat b001 = 4 beats b010 = 8 beats b011 = 16 beats b100 = 32 beats b101 = continuous b110-b111 = reserved. For a NAND memory interface these bits are reserved, and written as zero.
[2]	set_rd_sync	Contains the value to be written to the specific SRAM chip opmode Register rd_sync bit. Memory in sync mode when set. For a NAND memory interface this bit is reserved, and written as zero.
[1:0]	set_mw	Contains the value to be written to the specific chip opmode Register *memory width* (mw) field. Encodes the memory data bus width: b00 = 8 bits[b] b01 = 16 bits[b] b10 = 32 bits b11 = reserved. You can program this to the configured width, or half that width. See *Memory Interface Configuration Register* on page 3-9.

a. For asynchronous transfers:
- the SMC always aligns read bursts to the memory burst boundary, when set_rd_sync = 0
- the SMC always aligns write bursts to the memory burst boundary, when set_wr_sync = 0.

b. For a NAND interface, only 8-bit and 16-bit are valid settings.

3.3.8 Refresh Period 0 Register

The refresh_0 Register characteristics are:

Purpose Controls the insertion of idle cycles during consecutive bursts. This enables PSRAM devices on memory interface 0 to initiate a refresh cycle.

Usage constraints Not accessible in the Reset or Low-power states.

Configurations Available when memory interface 0 is type SRAM.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-13 shows the refresh_0 Register bit assignments.

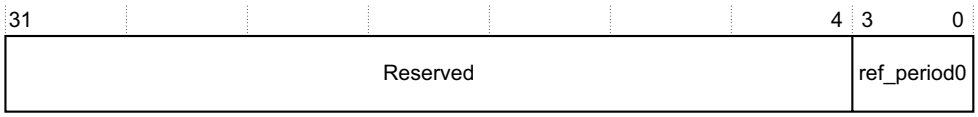


Figure 3-13 refresh_0 Register bit assignments

Table 3-9 shows the refresh_0 Register bit assignments.

Table 3-9 refresh_0 Register bit assignments

Bits	Name	Function
[31:4]	-	Reserved, read undefined, write as zero.
[3:0]	ref_period0	Sets the number of consecutive memory bursts[a] that the SMC permits, on memory interface 0, before it deasserts the chip select. The options are: b0000 = disables the insertion of idle cycles between consecutive bursts b0001 = an idle cycle occurs after each burst b0010 = an idle cycle occurs after 2 consecutive bursts b0011 = an idle cycle occurs after 3 consecutive bursts b0100 = an idle cycle occurs after 4 consecutive bursts . . . b1111 = an idle cycle occurs after 15 consecutive bursts.

a. In continuous mode the memory bursts are limited to 32 beats.

3.3.9 Refresh Period 1 Register

The refresh_1 Register characteristics are:

Purpose Controls the insertion of idle cycles during consecutive bursts. This enables PSRAM devices on memory interface 1 to initiate a refresh cycle.

Usage constraints Not accessible in the Reset or Low-power states.

Configurations Available when memory interface 1 is type SRAM.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-14 shows the refresh_1 Register bit assignments.

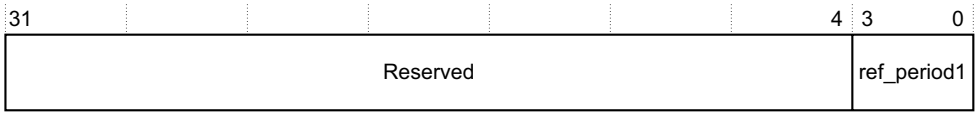


Figure 3-14 refresh_1 Register bit assignments

Table 3-10 shows the refresh_1 Register bit assignments.

Table 3-10 refresh_1 Register bit assignments

Bits	Name	Function
[31:4]	-	Reserved, read undefined, write as zero.
[3:0]	ref_period1	Sets the number of consecutive memory bursts[a] that the SMC permits, on memory interface 1, before it deasserts the chip select. The options are: b0000 = disables the insertion of idle cycles between consecutive bursts b0001 = an idle cycle occurs after each burst b0010 = an idle cycle occurs after 2 consecutive bursts b0011 = an idle cycle occurs after 3 consecutive bursts b0100 = an idle cycle occurs after 4 consecutive bursts . . . b1111 = an idle cycle occurs after 15 consecutive bursts.

a. In continuous mode the memory bursts are limited to 32 beats.

3.3.10 SRAM Cycles Register

The sram_cycles<x>_<n> Register characteristics are:

Purpose Returns the programmed timing parameters for SRAMs that connect to memory interface <x> and chip select <n>.

Usage constraints Not accessible in the Reset state.

Configurations Available when memory interface 0, or 1, is type SRAM.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-15 shows the sram_cycles<x>_<n> Register bit assignments.

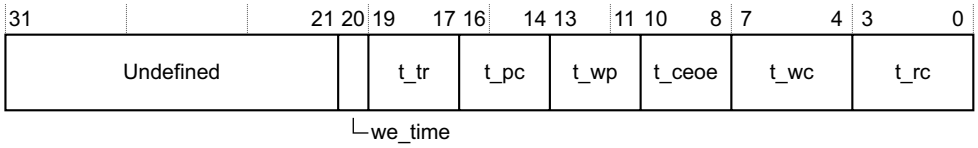


Figure 3-15 sram_cycles Register bit assignments

Table 3-11 shows the sram_cycles<x>_<n> Register bit assignments.

Table 3-11 sram_cycles Register bit assignments

Bits	Name	Function
[31:21]	-	Reserved, read undefined.
[20]	we_time<x>	For asynchronous multiplexed transfers this bit returns when the SMC asserts **we_n**: 0 = SMC asserts **we_n** two **mclk** cycles after asserting **cs_n**. See Figure 2-17 on page 2-29. 1 = SMC asserts **we_n** and **cs_n** together.
[19:17]	t_tr<x>	Returns the turnaround time. Minimum permitted value = 1.
[16:14]	t_pc<x>	Returns the page cycle time. Minimum permitted value = 1.

Table 3-11 sram_cycles Register bit assignments (continued)

Bits	Name	Function
[13:11]	t_wp<x>	Returns the **we_n** assertion delay. Minimum permitted value = 1.
[10:8]	t_ceoe<x>	Returns the **oe_n** assertion delay. Minimum permitted value = 1.
[7:4]	t_wc<x>	Returns the write cycle time. Minimum permitted value = 2.
[3:0]	t_rc<x>	Returns the read cycle time. Minimum permitted value = 2.

3.3.11 NAND Cycles Register

The nand_cycles<x>_<n> Register characteristics are:

Purpose Returns the programmed timing parameters for NANDs that connect to memory interface <x> and chip select <n>.

Usage constraints Not accessible in the Reset state.

Configurations Available when memory interface 0, or 1, is type NAND.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-16 shows the nand_cycles<x>_<n> Register bit assignments.

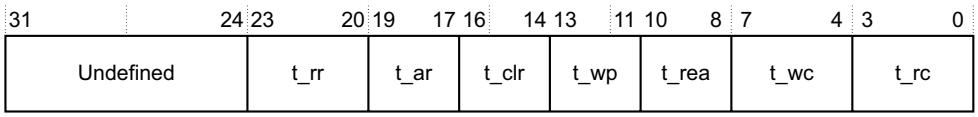


Figure 3-16 nand_cycles Register bit assignments

Table 3-12 shows the nand_cycles<x>_<n> Register bit assignments.

Table 3-12 nand_cycles Register bit assignments

Bits	Name	Function
[31:24]	-	Reserved, read undefined.
[23:20]	t_rr<x>	Returns the **busy** to **re_n** time. Minimum permitted value = 0.
[19:17]	t_ar<x>	Returns the ID read time. Minimum permitted value = 0.
[16:14]	t_clr<x>	Returns the status read time. Minimum permitted value = 0.
[13:11]	t_wp<x>	Returns the **we_n** deassertion delay. Minimum permitted value = 1.
[10:8]	t_rea<x>	Returns the **re_n** assertion delay. Minimum permitted value = 1.
[7:4]	t_wc<x>	Returns the write cycle time. Minimum permitted value = 2.
[3:0]	t_rc<x>	Returns the read cycle time. Minimum permitted value = 2.

3.3.12 Operating Mode Status Register

The opmode<x>_<n> Register characteristics are:

Purpose Returns the programmed operating mode for memory devices that connect to memory interface <x> and chip select <n>.

Usage constraints Not accessible in the Reset state.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-17 shows the opmode<x>_<n> Register bit assignments.

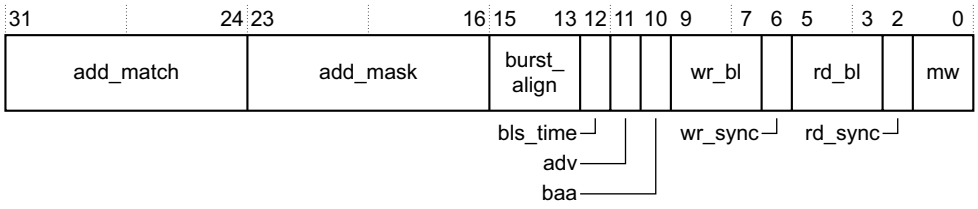


Figure 3-17 opmode Register bit assignments

Table 3-13 shows the opmode<x>_<n> Register bit assignments.

Table 3-13 opmode Register bit assignments

Bits	Name	Function
[31:24]	addr_match<x>	Returns the value of the **addr_match<x>_<n>[7:0]** tie-off. This is the comparison value for address bits [31:24] to determine the chip that is selected.
[23:16]	addr_mask<x>	Returns the value of the **addr_mask<x>_<n>[7:0]** tie-off. This is the mask for address bits[31:24]. A logic 1 indicates the bit is used for comparison.
[15:13]	burst_align<x>	When you program the SMC to perform synchronous transfers[a], these bits return the memory burst operating mode: b000 = bursts can cross any address boundary. This is the default setting. b001 = burst split on memory burst boundary, that is, 32 beats for continuous. b010 = burst split on 64 beat boundary. b011 = burst split on 128 beat boundary. b100 = burst split on 256 beat boundary. b101-b111 = reserved. The reset value is b000. For a NAND memory interface these bits are reserved.
[12]	bls_time<x>	Returns the byte lane strobe operating mode for an SRAM memory interface: 0 = bls timing equals chip select timing. This is the default setting. 1 = bls timing equals **we_n** timing. For a NAND memory interface this bit is reserved.
[11]	adv<x>	Returns the address advance signal operating mode for an SRAM memory interface: 0 = SMC ties **adv_n** HIGH. This is the default setting. 1 = SMC sets **adv_n** LOW at the start of a transfer. For a NAND memory interface this bit is reserved.
[10]	baa<x>	Returns the burst address advance signal operating mode for an SRAM memory interface: 0 = SMC ties **baa_n** HIGH. This is the default setting. 1 = SMC sets **baa_n** LOW. For a NAND memory interface this bit is reserved.

Table 3-13 opmode Register bit assignments (continued)

Bits	Name	Function
[9:7]	wr_bl<x>	Returns the memory burst length for writes on an SRAM memory interface: b000 = 1 beat. This is the default setting. b001 = 4 beats b010 = 8 beats b011 = 16 beats b100 = 32 beats b101 = continuous b110-b111 = reserved. For a NAND memory interface these bits are reserved.
[6]	wr_sync<x>	Returns the write operating mode for an SRAM memory interface: 0 = SMC performs asynchronous writes. This is the default setting. 1 = SMC performs synchronous writes. For a NAND memory interface this bit is reserved.
[5:3]	rd_bl<x>	Returns the memory burst length for reads on an SRAM memory interface: b000 = 1 beat. This is the default setting. b001 = 4 beats b010 = 8 beats b011 = 16 beats b100 = 32 beats b101 = continuous b110-b111 = reserved. For a NAND memory interface these bits are reserved.
[2]	rd_sync<x>	Returns the read operating mode for an SRAM memory interface: 0 = SMC performs asynchronous reads. This is the default setting. 1 = SMC performs synchronous reads. For a NAND memory interface this bit is reserved.
[1:0]	mw<x>	Returns the SMC memory data bus width: b00 = 8 bits b01 = 16 bits b10 = 32 bits b11 = reserved. You can use the **sram_mw_<x>[1:0]** or **nand_mw_<x>** tie-offs to set the memory width for chip select 0, memory interface <x>, to enable booting from that chip. The reset value of memory width for all other chip selects is the configured width.

a. For asynchronous transfers:
- the SMC always aligns read bursts to the memory burst boundary, when rd_sync = 0
- the SMC always aligns write bursts to the memory burst boundary, when wr_sync = 0.

3.3.13 User Status Register

The user_status Register characteristics are:

Purpose Provides the status of the **user_status[7:0]** inputs.

Usage constraints No usage constraints.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-18 shows the user_status Register bit assignments.

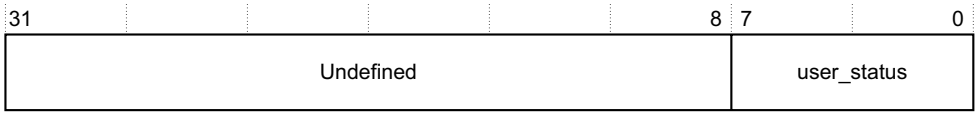


Figure 3-18 user_status Register bit assignments

Table 3-14 shows the user_status Register bit assignments.

Table 3-14 user_status Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	user_status	This value returns the state of the **user_status[7:0]** inputs

3.3.14 User Config Register

The user_config Register characteristics are:

Purpose Controls the state of the **user_config[7:0]** outputs.

Usage constraints No usage constraints.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-19 shows the user_config Register bit assignments.

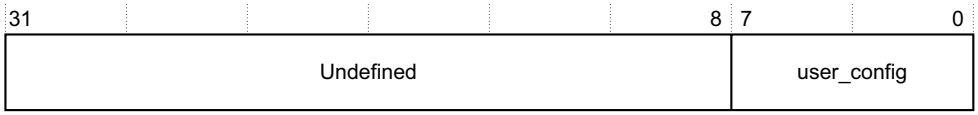


Figure 3-19 user_config Register bit assignments

Table 3-15 shows the user_config Register bit assignments.

Table 3-15 user_config Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, write as zero
[7:0]	user_config	This value sets the state of the **user_config[7:0]** outputs

3.3.15 ECC Status Register

The ecc<x>_status Register characteristics are:

Purpose For memory interface <x>, use this to:
- obtain the status of the ECC block
- clear the ECC interrupts.

Usage constraints
- Not accessible in the Reset state.

- You can write to the ecc<x>_int_status field but the SMC ignores writes to all other fields.

Configurations Available when an SMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-20 shows the ecc<x>_status Register bit assignments.

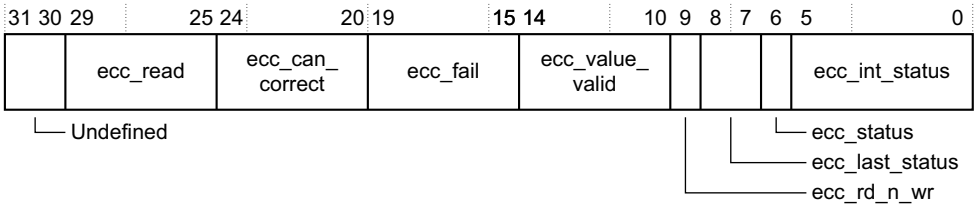


Figure 3-20 ecc<x>_status Register bit assignments

Table 3-16 shows the ecc<x>_status Register bit assignments.

Table 3-16 ecc<x>_status Register bit assignments

Bits	Name	Function
[31:30]	-	Reserved, read undefined.
[29:25]	ecc_read	Read flags for ECC blocks. Indicates whether the stored ECC value for each block has been read from memory 0 = not read 1 = read.
[24:20]	ecc_can_correct	Correctable flag for each ECC block. Indicates if the detected error is correctable[a]: 0 = not correctable 1 = correctable. See Table 2-33 on page 2-48 for the decoding information of the ecc_can_correct flag and ecc_fail flag.
[19:15]	ecc_fail	Pass, fail flag for each ECC block: 0 = pass 1 = fail. See Table 2-33 on page 2-48 for the decoding information of the ecc_can_correct flag and ecc_fail flag.
[14:10]	ecc_value_valid	Valid flag for each ECC block.
[9]	ecc_rd_n_wr	0 = write 1 = read.

Table 3-16 ecc<x>_status Register bit assignments (continued)

Bits	Name	Function
[8:7]	ecc_last_status	b00 = Completed successfully b01 = Unaligned address, or out-of-range b10 = Data stop after incomplete block b11 = Data stopped but values not read/written because of ecc_jump value. **Note** The ecc_last_status bit is only updated at the completion of an ECC calculation.
[6]	ecc_status	Provides the status of the ECC block: 0 = idle 1 = busy.
	ecc_int_status	The interrupts are:
[5]		Abort.
[4]		Extra block (if used).
[3]		Block 3.
[2]		Block 2.
[1]		Block 1.
[0]		Block 0.
		Note To clear an interrupt, write a 1 to the appropriate bit.

a. The SMC detects but does not correct errors. See *Correcting errors* on page 2-48 for more information.

3.3.16 ECC Configuration Register

The ecc<x>_cfg Register characteristics are:

Purpose Configures the ECC block for memory interface <x>. See *Error Correction Code block* on page 2-42 for more information about the settings in this register.

Usage constraints
- Not accessible in the Reset state.
- Some bits are only available when the ECC Extra Block Enable option is configured.

Note

You must not write to this register when the ECC block is busy. You can read the current ECC block status from the *ECC Status Register* on page 3-24.

Configurations Available when an SMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-21 on page 3-27 shows the ecc<x>_cfg Register bit assignments.

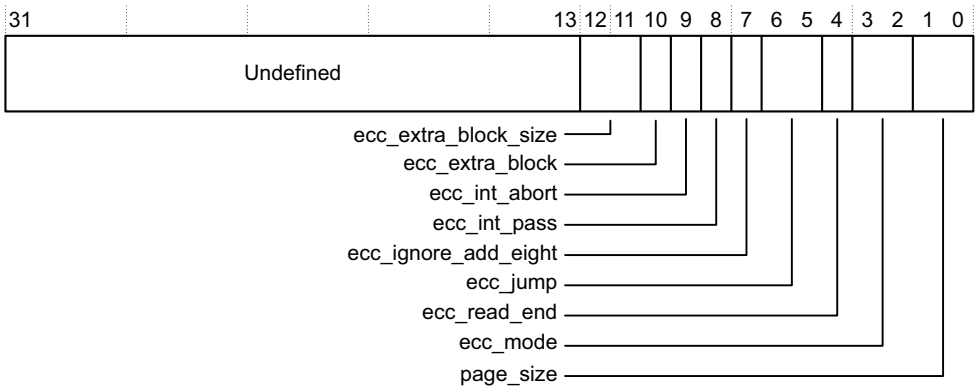


Figure 3-21 ecc<x>_cfg Register bit assignments

Table 3-17 shows the ecc<x>_cfg Register bit assignments.

Table 3-17 ecc<x>_cfg Register bit assignments

Bits	Name	Function
[31:13]	-	Reserved, read undefined
[12:11]	ecc_extra_block_size	The size of the extra block in memory after the last 512-byte block: b00 = 4 bytes b01 = 8 bytes b10 = 16 bytes b11 = 32 bytes. **Note** These bits are only present if you configure the SMC to use the ECC Extra Block Enable option. See the *CoreLink SMC-35x AXI Static Memory Controller Series Supplement to AMBA Designer (FD001) User Guide* for information about enabling this option.
[10]	ecc_extra_block	If configured, this enables a small block for extra information after the last 512 bytes block in the page. **Note** These bits are only present if the ECC Extra Block Enable option is configured.
[9]	ecc_int_abort	Interrupt on ECC abort.
[8]	ecc_int_pass	Interrupt when a correct ECC value is read from memory.
[7]	ecc_ignore_add_eight	This bit is used to indicate if A8 is output with the address, required to find the aligned start of blocks: 0 = A8 is output 1 = A8 is not output. See *Secondary mode addressing* on page 2-45.
[6:5]	ecc_jump	Indicates that the memory supports column change address commands: b00 = no jumping, reads and writes only occur at end of page b01 = jump using column change commands b10 = jump using full command b11 = reserved.

Table 3-17 ecc<x>_cfg Register bit assignments (continued)

Bits	Name	Function
[4]	ecc_read_end	Indicates when ECC values are read from memory: 0 = the ECC value for a block must be read immediately after the block. Data access must stop on a 512 byte boundary. 1 = ECC values for all blocks are read at the end of the page.
[3:2]	ecc_mode	This specifies the mode of the ECC block: b00 = bypassed. b01 = ECC values are calculated and made available on the APB interface, but they are not read from or written to memory. b10 = ECC values and calculated and read/written to memory. For a read, the ECC value is checked and the result of the check is made available on the APB interface. b11 = reserved.
[1:0]	page_size	The number of 512 byte blocks in a page: b00 = No 512 byte blocks. Reserved if an ecc_extra_block is not configured, or if an ecc_extra_block is configured but is not enabled. b01 = One 512 byte block. b10 = Two 512 byte blocks. b11 = Four 512 byte blocks.

3.3.17 ECC Command 0 Register

The ecc<x>_memcmd0 Register characteristics are:

Purpose Contains the commands that the ECC block uses to detect the start of an ECC operation for memory interface <x>.

Usage constraints Not accessible in the Reset state.

Configurations Available when an SMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-22 shows the ecc<x>_memcmd0 Register bit assignments.

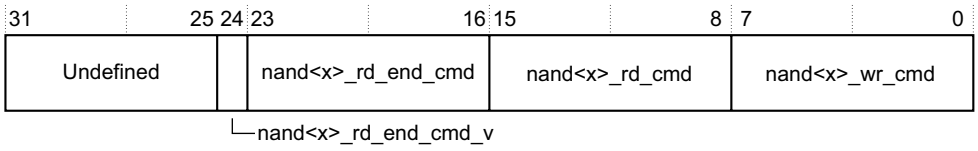


Figure 3-22 ecc<x>_memcmd0 Register bit assignments

Table 3-18 shows the ecc<x>_memcmd0 Register bit assignments.

Table 3-18 ecc<x>_memcmd0 Register bit assignments

Bits	Name	Function
[31:25]	-	Reserved, undefined, write as zero
[24]	nand<x>_rd_end_cmd_v	Use end command

Table 3-18 ecc<x>_memcmd0 Register bit assignments (continued)

Bits	Name	Function
[23:16]	nand<x>_rd_end_cmd	The NAND command to indicate the end of a read (0x30)
[15:8]	nand<x>_rd_cmd	The NAND command to initiate a read (0x00)
[7:0]	nand<x>_wr_cmd	The NAND command to initiate a write (0x80)

3.3.18 ECC Command 1 Register

The ecc<x>_memcmd1 Register characteristics are:

Purpose Contains the commands that the ECC block uses to access different parts of a NAND page, for memory interface <x>. The reset value is suitable for *Open NAND Flash Interface* (ONFi) 1.0 compliant devices.

Usage constraints Not accessible in the Reset state.

Configurations Available when an SMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-23 shows the ecc<x>_memcmd1 Register bit assignments.

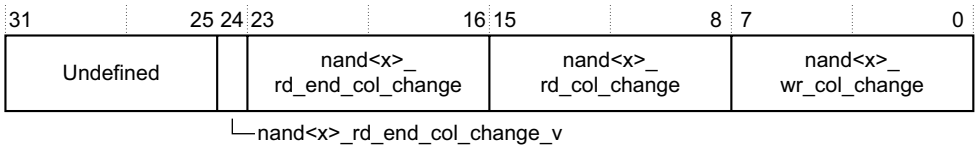


Figure 3-23 ecc<x>_memcmd1 Register bit assignments

Table 3-19 shows the ecc<x>_memcmd1 Register bit assignments.

Table 3-19 ecc<x>_memcmd1 Register bit assignments

Bits	Name	Function
[31:25]	-	Reserved, undefined, write as zero
[24]	nand<x>_rd_end_col_change_v	Use end command
[23:16]	nand<x>_rd_end_col_change	The NAND command to indicate the end of a column change read (0xE0)
[15:8]	nand<x>_rd_col_change	The NAND command (0x05) that is either: • a command to initiate a column change read • a spare bits pointer command.
[7:0]	nand<x>_wr_col_change	The NAND command to initiate a column change write (0x85)

3.3.19 ECC Address 0 Register

The ecc<x>_addr0 Register characteristics are:

Purpose Returns the lower 32 bits of the ECC address, for memory interface <x>.

Usage constraints Not accessible in the Reset state.

Configurations Available when an SMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-24 shows the ecc<x>_addr0 Register bit assignments.

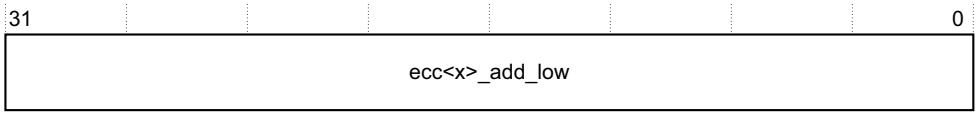


Figure 3-24 ecc<x>_addr0 Register bit assignments

Table 3-20 shows the ecc<x>_addr0 Register bit assignments.

Table 3-20 ecc<x>_addr0 Register bit assignments

Bits	Name	Function
[31:0]	ecc<x>_add_low	Address bits 31 to 0 for memory interface <x>

3.3.20 ECC Address 1 Register

The ecc<x>_addr1 Register characteristics are:

Purpose Returns the upper 24 bits of the ECC address, for memory interface <x>.

Usage constraints Not accessible in the Reset state.

Configurations Available when an SMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-25 shows the ecc<x>_addr1 Register bit assignments.

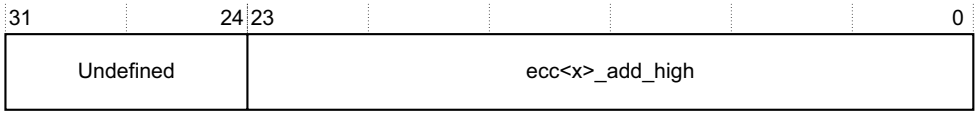


Figure 3-25 ecc<x>_addr1 Register bit assignments

Table 3-21 shows the ecc<x>_addr1 Register bit assignments.

Table 3-21 ecc<x>_addr1 Register bit assignments

Bits	Name	Function
[31:24]	-	Reserved, read undefined
[23:0]	ecc<x>_add_high	Address bits [55:32] for memory interface <x>

3.3.21 ECC Block Registers

The ecc<x>_block<3:0> Register characteristics are:

Purpose Each of the four registers return ECC block information for memory interface <x>.

Usage constraints Not accessible in the Reset state.

Configurations Available when an SMC is configured to support ECC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-26 shows the ecc<x>_block<3:0> Register bit assignments.

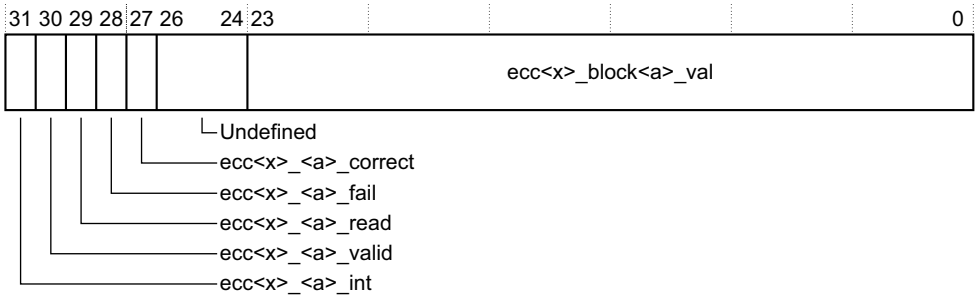


Figure 3-26 ecc<x>_block<3:0> Register bit assignments

Table 3-22 shows the ecc<x>_block<3:0> Register bit assignments.

Table 3-22 ecc<x>_block<3:0> Register bit assignments

Bits	Name	Function
[31]	ecc<x>_<a>_int	Interrupt flag for block <a>. To clear this bit, write any value to the register.
[30]	ecc<x>_<a>_valid	Indicates if the ECC value for block <a> is valid.
[29]	ecc<x>_<a>_read	Indicates if the ECC value for block <a> has been read from memory.
[28]	ecc<x>_<a>_fail	Indicates if the ECC value for block <a> has failed.See Table 2-33 on page 2-48 for the decoding information of the ecc_can_correct flag and ecc_fail flag.
[27]	ecc<x>_<a>_correct	Indicates if block <a> is correctable.See Table 2-33 on page 2-48 for the decoding information of the ecc_can_correct flag and ecc_fail flag.
[26:24]	-	Reserved, read undefined, write as zero.
[23:0]	ecc<x>_block<a>_val	ECC value of check result for block <a>.

3.3.22 ECC Extra Block Register

The ecc<x>_extra_block Register characteristics are:

Purpose Returns ECC extra block information for memory interface <x>.

Usage constraints Not accessible in the Reset state.

Configurations Available when an SMC is configured to support an ECC extra block.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-27 on page 3-32 shows the ecc<x>_extra_block Register bit assignments.

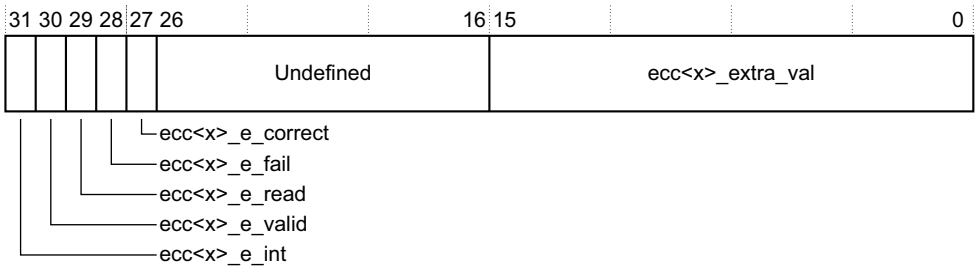


Figure 3-27 ecc<x>_extra_block Register bit assignments

Table 3-23 shows the ecc<x>_extra_block Register bit assignments.

Table 3-23 ecc<x>_extra_block Register bit assignments

Bits	Name	Function
[31]	ecc<x>_e_int	Interrupt flag for the extra block. To clear this bit, write any value to the register.
[30]	ecc<x>_e_valid	Indicates if the ECC value for extra block is valid.
[29]	ecc<x>_e_read	Indicates if the ECC value for extra block has been read from memory.
[28]	ecc<x>_e_fail	Indicates if the ECC value for extra block has failed.
[27]	ecc<x>_e_correct	Indicates if the extra block is correctable.
[26:24]	-	Reserved, read undefined, write as zero.
[23:0]	ecc<x>_extra_val	ECC value of check result for the extra block.

3.3.23 Peripheral Identification Registers 0-3

The periph_id_[3:0] Register characteristics are:

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

Usage constraints Not accessible in the Reset state.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

These registers can conceptually be treated as a single register that holds a 32-bit peripheral ID value. Figure 3-28 shows the correspondence between bits [7:0] of the periph_id Registers and the conceptual 32-bit Peripheral ID Register.

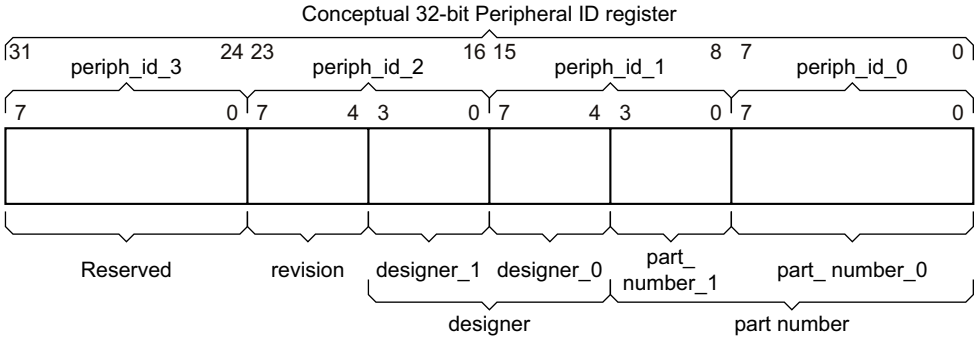


Figure 3-28 periph_id_[3:0] Register bit assignments

Table 3-24 shows the register bit assignments for the conceptual 32-bit peripheral ID register.

Table 3-24 Conceptual peripheral ID register bit assignments

Bits	Name	Description
[31:25]	-	Reserved, read undefined.
[24]	integration_cfg	Configuration option is peripheral-specific. See *Peripheral Identification Register 3* on page 3-34.
[23:20]	revision	Identifies the RTL revision of the peripheral.
[19:12]	designer	Identifies the designer. This is `0x41` for ARM.
[11:0]	part_number	Identifies the peripheral. The part numbers for the SMC are: • `0x351` for SMC-351 • `0x352` for SMC-352 • `0x353` for SMC-353 • `0x354` for SMC-354.

The following sections describe the periph_id Registers:

- *Peripheral Identification Register 0*
- *Peripheral Identification Register 1*
- *Peripheral Identification Register 2* on page 3-34
- *Peripheral Identification Register 3* on page 3-34.

Peripheral Identification Register 0

The periph_id_0 Register is hard-coded and the fields in the register determine the reset value. Table 3-25 shows the register bit assignments.

Table 3-25 periph_id_0 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	part_number_0	Returns `0x5x`, see Table 3-24

Peripheral Identification Register 1

The periph_id_1 Register is hard-coded and the fields in the register determine the reset value. Table 3-26 shows the register bit assignments.

Table 3-26 periph_id_1 Register bit assignments

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

Peripheral Identification Register 2

The periph_id_2 Register is hard-coded and the fields in the register determine the reset value. Table 3-27 shows the register bit assignments.

Table 3-27 periph_id_2 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:4]	revision	These bits read back as: • 0x1 for r1p0 • 0x2 for r1p1 • 0x3 for r1p2 • 0x4 for r2p0 • 0x5 for r2p1 • 0x6 for r2p2
[3:0]	designer_1	Returns 0x4

Peripheral Identification Register 3

The periph_id_3 Register is hard-coded and the fields in the register determine the reset value. Table 3-28 shows the register bit assignments.

Table 3-28 periph_id_3 Register bit assignments

Bits	Name	Function
[31:0]	-	Reserved, read undefined.

3.3.24 CoreLink Identification Registers 0-3

The pcell_id_[3:0] Register characteristics are:

Purpose When concatenated, these four registers return 0xB105F00D to indicate that the SMC is a CoreLink peripheral.

Usage constraints Not accessible in the Reset state.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 3-1 on page 3-5.

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

Figure 3-29 on page 3-35 shows the correspondence between bits [7:0] of the pcell_id Registers and the conceptual 32-bit CoreLink ID Register.

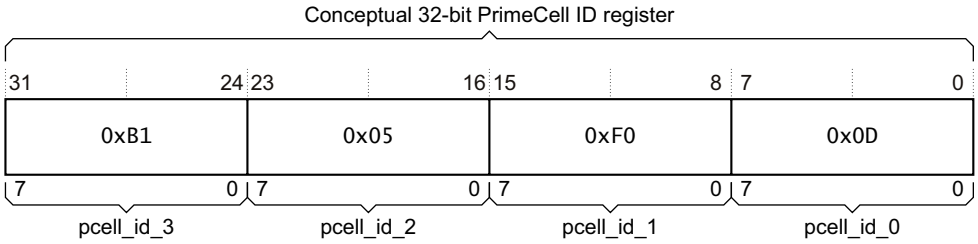


Figure 3-29 pcell_id Register bit assignments

Table 3-29 shows the register bit assignments.

Table 3-29 pcell_id Register bit assignments

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

The following subsections describe the identification registers:

- *CoreLink Identification Register 0*
- *CoreLink Identification Register 1* on page 3-36
- *CoreLink Identification Register 2* on page 3-36
- *CoreLink Identification Register 3* on page 3-36.

Note

You cannot read these registers in the Reset state.

CoreLink Identification Register 0

The pcell_id_0 Register is hard-coded and the fields in the register determine the reset value. Table 3-30 shows the register bit assignments.

Table 3-30 pcell_id_0 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	pcell_id_0	These bits read back as 0x0D

CoreLink Identification Register 1

The pcell_id_1 Register is hard-coded and the fields in the register determine the reset value. Table 3-31 shows the register bit assignments.

Table 3-31 pcell_id_1 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	pcell_id_1	These bits read back as `0xF0`

CoreLink Identification Register 2

The pcell_id_2 Register is hard-coded and the fields in the register determine the reset value. Table 3-32 shows the register bit assignments.

Table 3-32 pcell_id_2 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	pcell_id_2	These bits read back as `0x5`

CoreLink Identification Register 3

The pcell_id_3 Register is hard-coded and the fields in the register determine the reset value. Table 3-33 shows the register bit assignments.

Table 3-33 pcell_id_3 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	pcell_id_3	These bits read back as `0xB1`

Chapter 4
Programmers Model for Test

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

4.1 Integration test registers

Test registers are provided for integration testing.

Figure 4-1 shows the integration test register map.

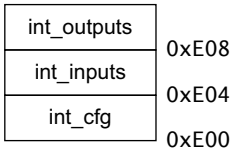


Figure 4-1 Integration test register map

Table 4-1 shows the integration test registers.

Table 4-1 SMC test register summary

Offset	Name	Type	Reset	Description
0xE00	int_cfg	R/W	0x0	*Integration Configuration Register* on page 4-3
0xE04	int_inputs	RO	-[a]	*Integration Inputs Register* on page 4-3
0xE08	int_outputs	WO	-	*Integration Outputs Register* on page 4-4

a. Dependent on the state of the tie-off signals.

Note

Signals that this section describes that end in zero are always valid. These signals reference interface 0.

Signals that this section describes that end in a one are only valid if configured for variants SMC-353 and SMC-354. These signals reference interface 1.

4.1.1 Integration Configuration Register

The int_cfg Register characteristics are:

Purpose Controls the enabling of the integration test logic.

Usage constraints Not accessible in the Reset state. ARM recommends that it is only accessed for integration testing or production testing.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 4-1 on page 4-2.

Figure 4-2 shows the int_cfg Register bit assignments.

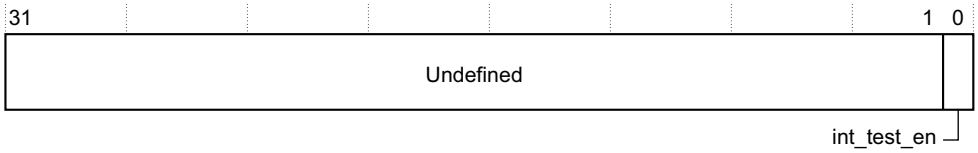


Figure 4-2 int_cfg Register bit assignments

Table 4-2 shows the int_cfg Register bit assignments.

Table 4-2 int_cfg Register bit assignments

Bits	Name	Function
[31:1]	-	Read undefined, write as zero.
[0]	int_test_en	Enables the integration test logic: 0 = disables the integration test logic 1 = enables the integration test logic. **Note** When the controller exits integration test mode, the **csysack** signal must be HIGH, even if the SoC does not use this signal. To satisfy this requirement you must program the int_outputs Register. See *Integration Outputs Register* on page 4-4.

4.1.2 Integration Inputs Register

The int_inputs Register characteristics are:

Purpose Provides the status of the following inputs:
- **async[1:0]**, **msync[1:0]**
- **ebibackoff[1:0]**, **ebigrant[1:0]**, and **use_ebi**.
- **csysreq**.

Usage constraints
- Not accessible in the Reset state.
- Integration test logic must be enabled otherwise it ignores writes and reads return `0x0`. To enable the integration test logic see *Integration Configuration Register*.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 4-1 on page 4-2.

Figure 4-3 on page 4-4 shows the int_inputs Register bit assignments.

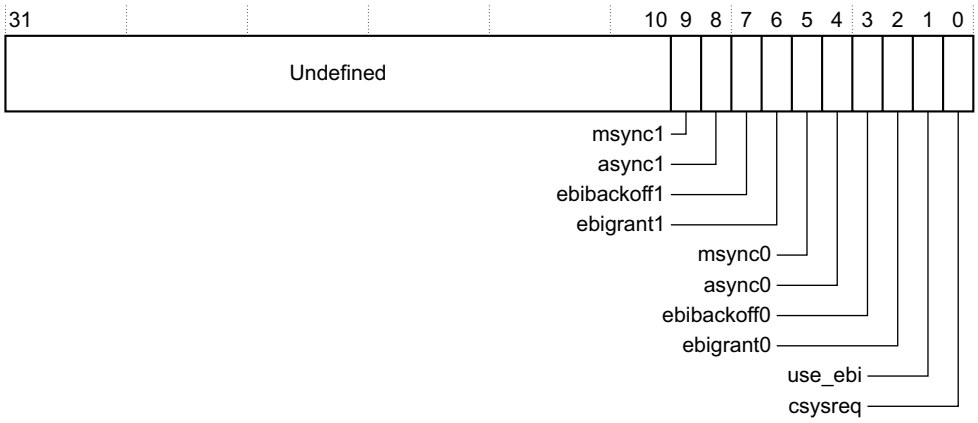


Figure 4-3 Int_inputs Register bit assignments

Table 4-3 shows the int_inputs Register bit assignments.

Table 4-3 Int_inputs Register bit assignments

Bits	Name	Function
[31:10]	-	Read undefined
[9]	msync1	Returns the status of the **msync1** input
[8]	async1	Returns the status of the **async1** input
[7]	ebibackoff1	Returns the status of the **ebibackoff1** input
[6]	ebigrant1	Returns the status of the **ebigrant1** input
[5]	msync0	Returns the status of the **msync0** input
[4]	async0	Returns the status of the **async0** input
[3]	ebibackoff0	Returns the status of the **ebibackoff0** input
[2]	ebigrant0	Returns the status of the **ebigrant0** input
[1]	use_ebi	Returns the status of the **use_ebi** input
[0]	csysreq	Returns the status of the **csysreq** input

4.1.3 Integration Outputs Register

The int_outputs Register characteristics are:

Purpose Enables an external master to control the state of the following outputs:

- **cactive**
- **csysack**
- **ebireq[1:0]**
- **smc_int**, **smc_int[1:0]**
- **ecc_int[1:0]**.

Usage constraints

- Not accessible in the Reset state.
- Some bits are only present when the SMC is configured to support *Error Correction Code* (ECC).

- Integration test logic must be enabled otherwise it ignores writes and reads return `0x0`. To enable the integration test logic see *Integration Configuration Register* on page 4-3.

Configurations Available in all configurations of the SMC.

Attributes See the register summary in Table 4-1 on page 4-2.

Figure 4-4 shows the int_outputs Register bit assignments.

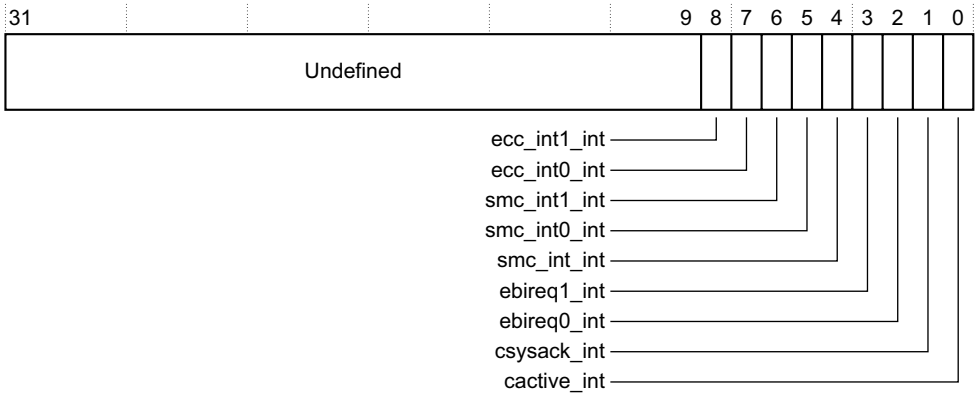


Figure 4-4 int_outputs Register bit assignments

Table 4-4 shows the int_outputs Register bit assignments.

Table 4-4 int_outputs Register bit assignments

Bits	Name	Function
[31:9]	-	Undefined, write as zero
[8]	ecc_int1_int	Controls the state of the **ecc_int1** output if the SMC supports ECC
[7]	ecc_int0_int	Controls the state of the **ecc_int0** output if the SMC supports ECC
[6]	smc_int1_int	Controls the state of the **smc_int1** output
[5]	smc_int0_int	Controls the state of the **smc_int0** output
[4]	smc_int_int	Controls the state of the **smc_int** output
[3]	ebireq1_int	Controls the state of the **ebireq1** output
[2]	ebireq0_int	Controls the state of the **ebireq0** output
[1]	csysack_int	Controls the state of the **csysack** output
[0]	cactive_int	Controls the state of the **cactive** output

Chapter 5
Device Driver Requirements

This chapter contains flow diagrams to aid in the development of a software driver for the SMC. It contains the following sections:

5.1 Memory initialization

Figure 5-1 and Figure 5-2 on page 5-3 show the sequence of events that a device driver must carry out to initialize the SMC and a memory device to ensure the configuration of both is synchronized.

Typically, PSRAM devices can have the mode register programmed using the address bus only. NOR flash memory devices are examples of memory that require mode register accesses to be carried out using a sequence of accesses using the address and data buses. Check the data sheet for the specific memory device you are configuring to determine the configuration method.

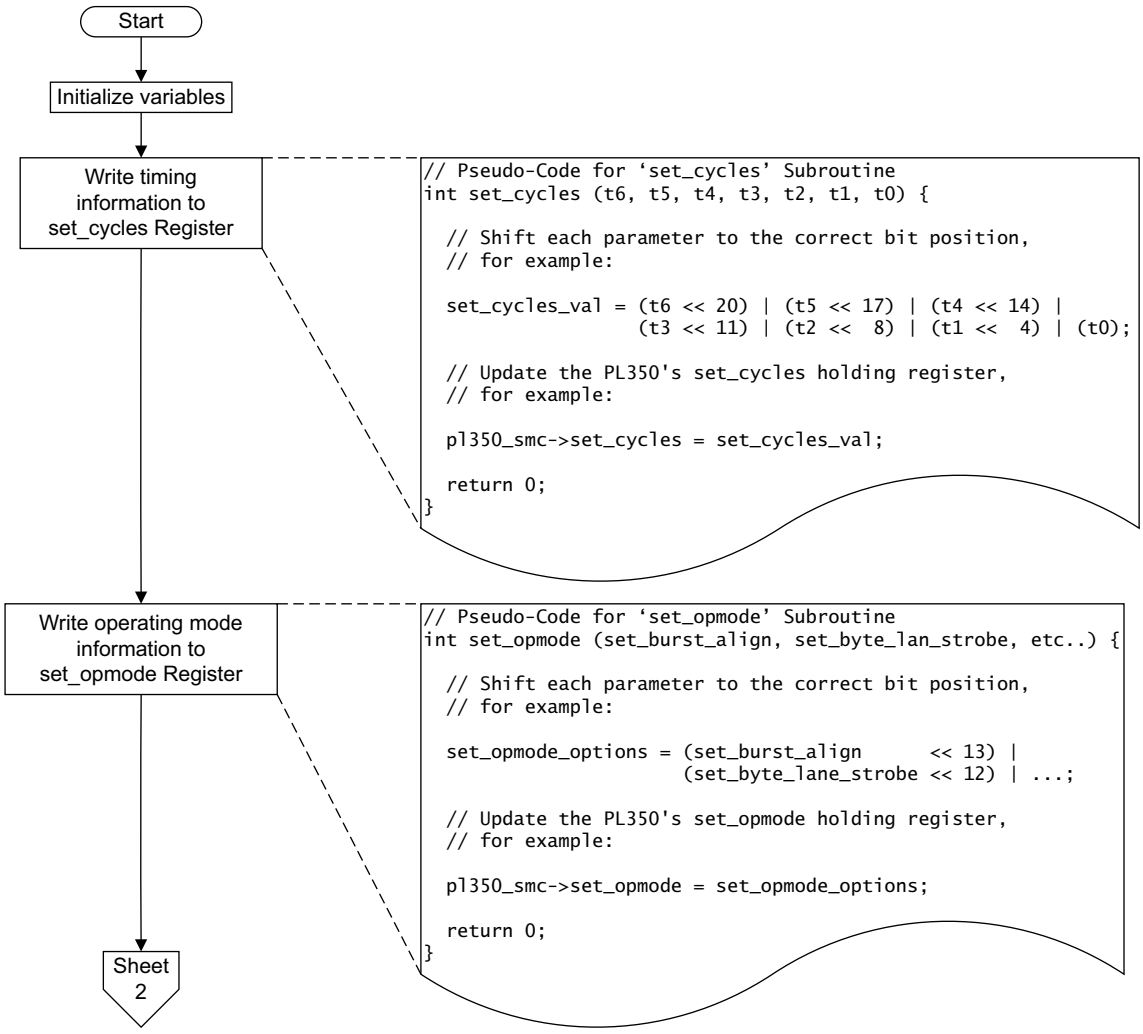


Figure 5-1 SMC and memory initialization sheet 1 of 2

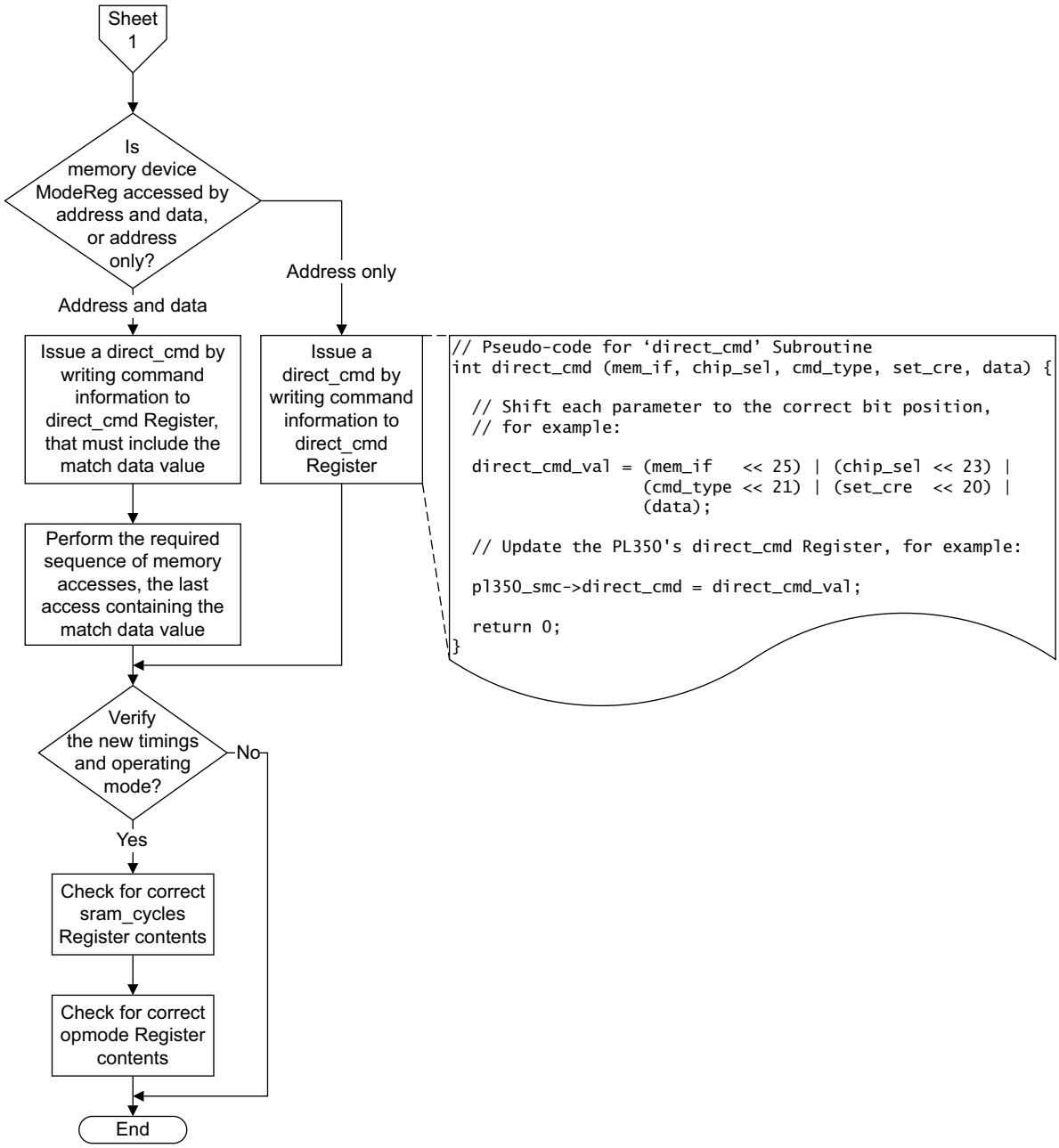


Figure 5-2 SMC and memory initialization sheet 2 of 2

5.2 NAND transactions

The transactions for NAND devices require some specific manipulation to format the transaction into the correct format for the SMC to map the transaction correctly to the NAND device.

The following figures show how a device driver is required to format the NAND command for the SMC:

- Figure 5-3 to Figure 5-4 on page 5-5
- Figure 5-5 on page 5-6 to Figure 5-6 on page 5-7.

For a command phase access, the NAND memory address is passed as data, see Table 2-1 on page 2-13 for a definition of how the address must be formatted. For a data phase access, the data is passed on the data bus.

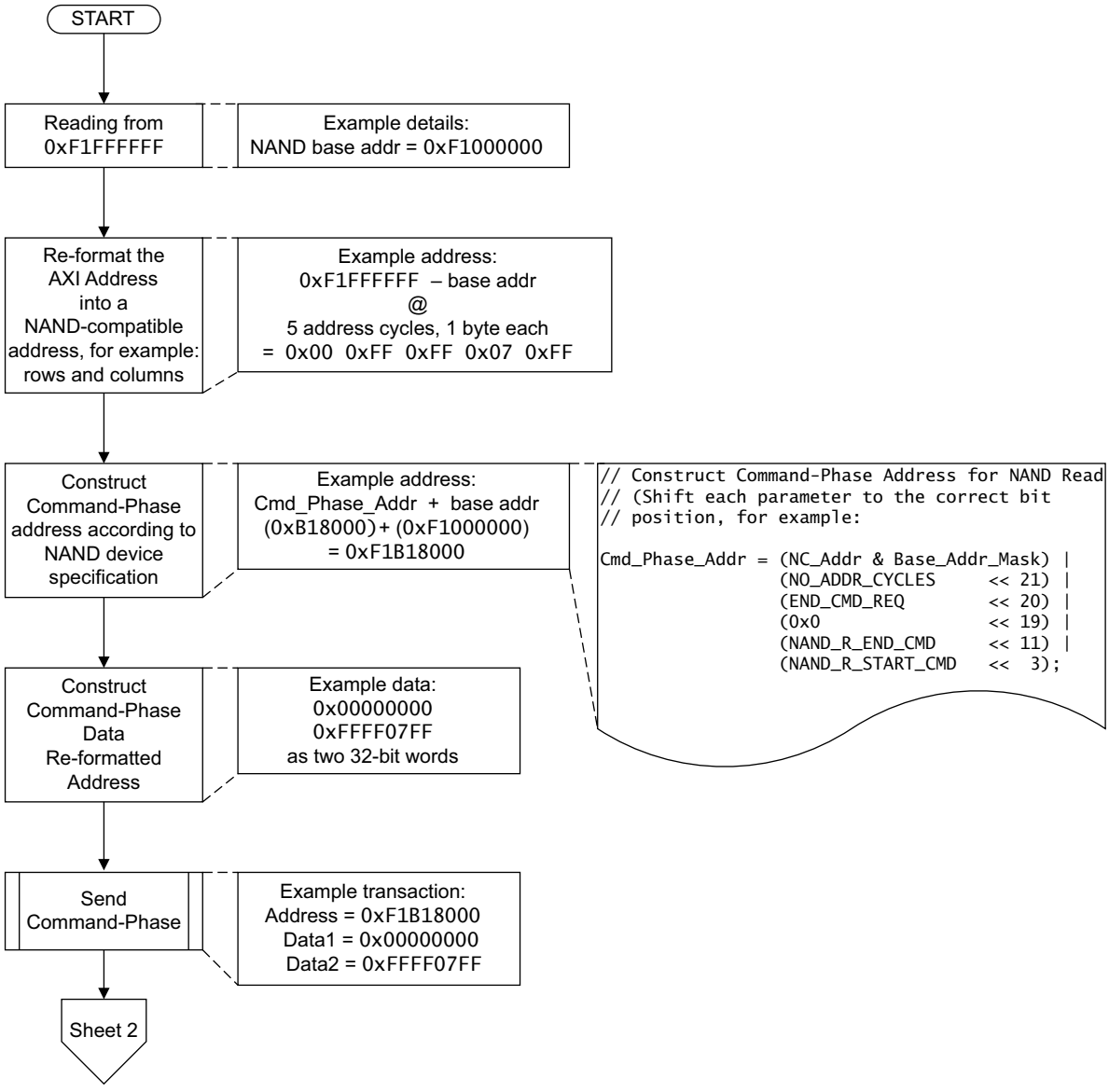


Figure 5-3 NAND read sheet 1 of 2

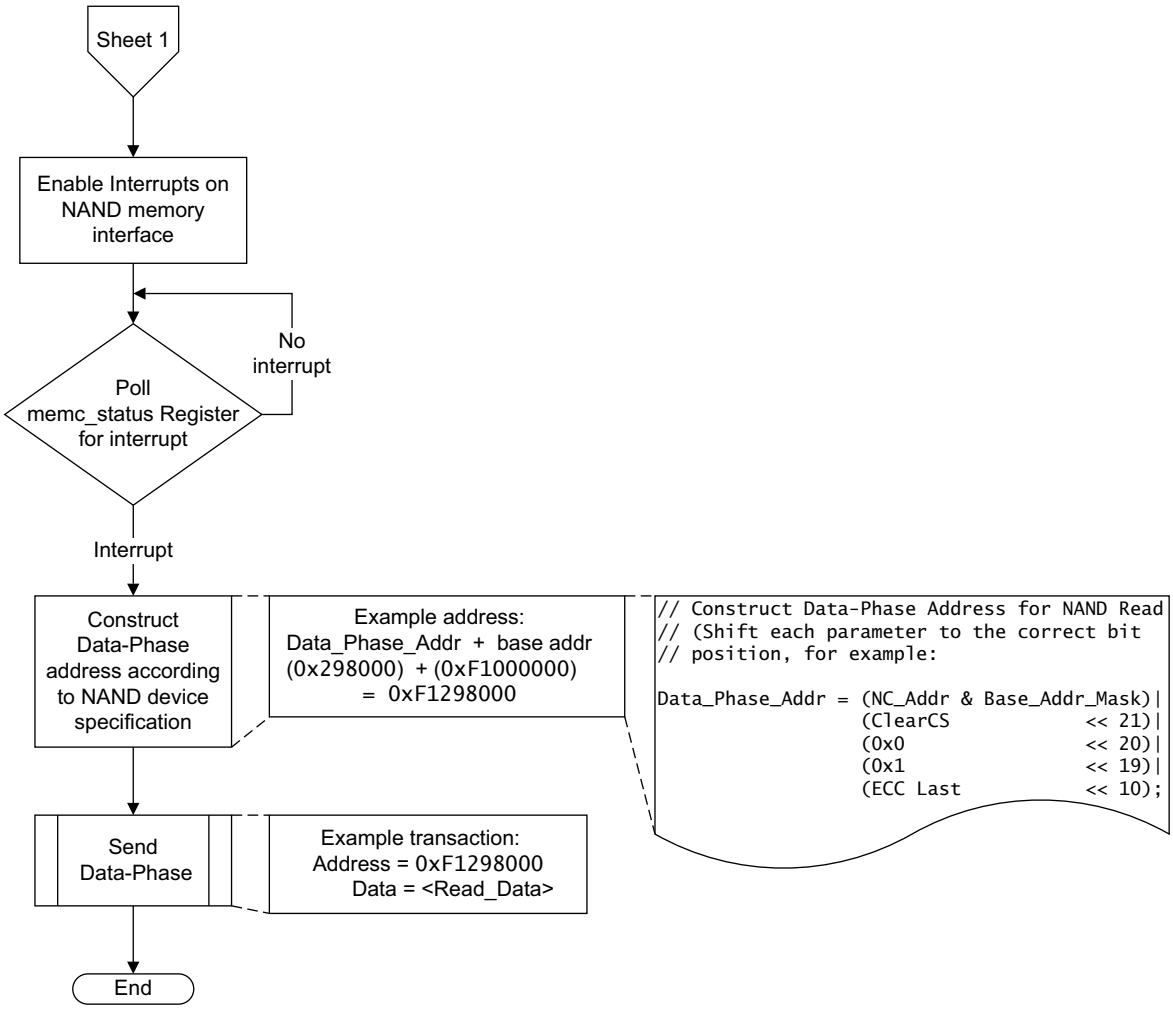


Figure 5-4 NAND read sheet 2 of 2

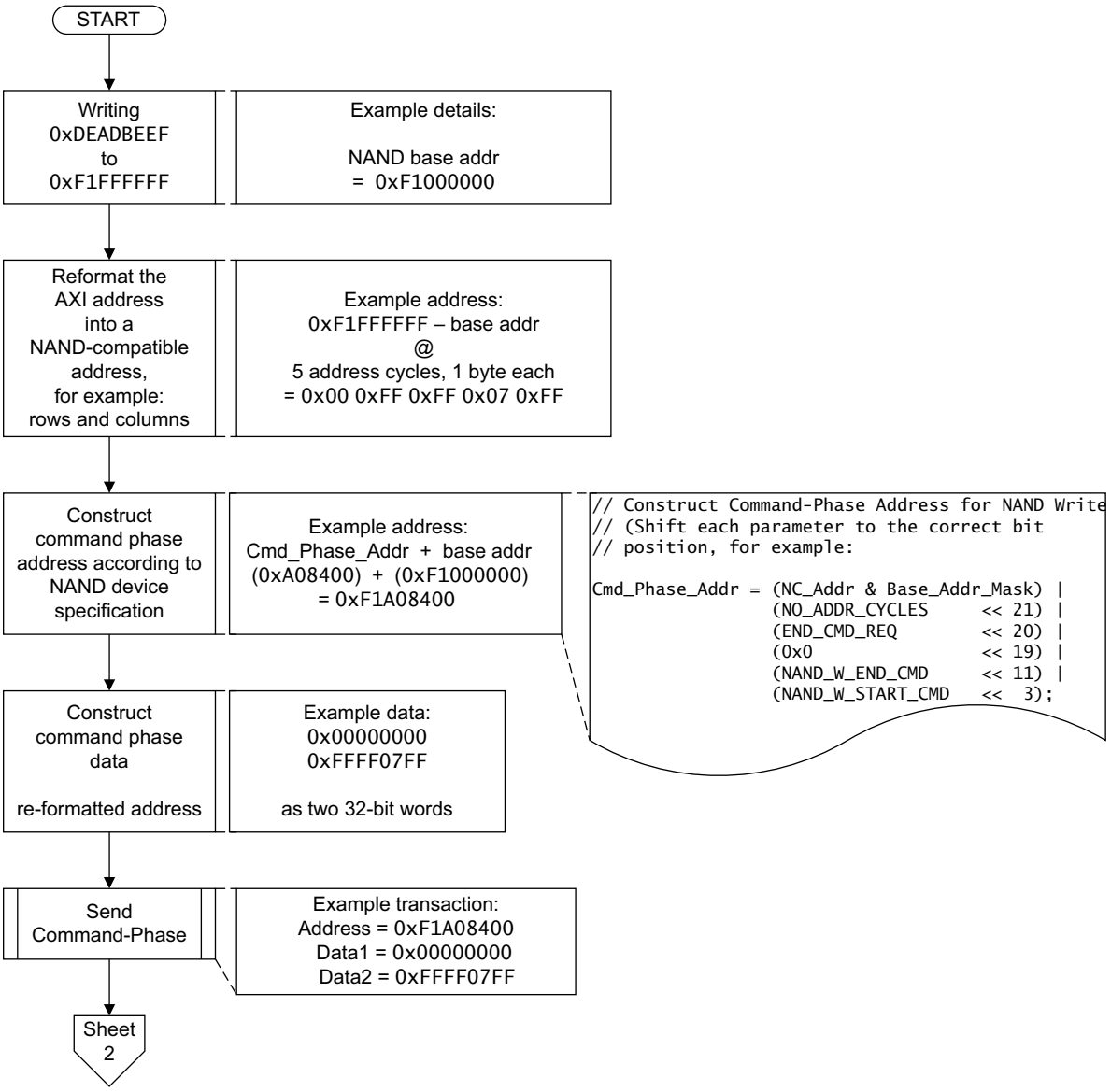


Figure 5-5 NAND write sheet 1 of 2

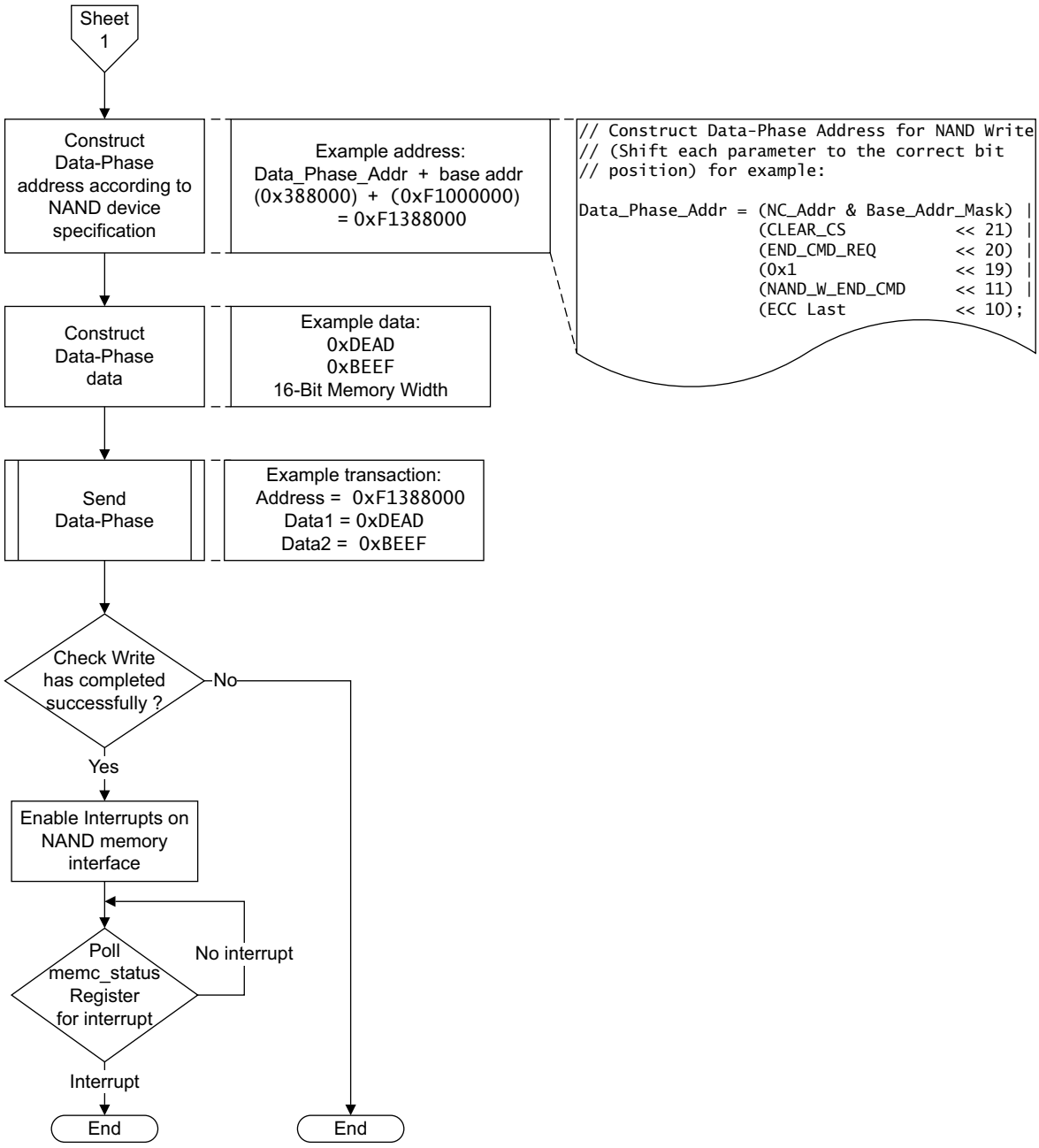


Figure 5-6 NAND write sheet 2 of 2

Chapter 6
Configurations

This chapter describes the four possible configurations of the SMC-35x series. The configurations are fixed in the number of memory interfaces and memory types supported, but configurable to support different numbers of chip selects and data bus widths. It contains the following sections:

Note

References are made to chapters and the signals appendix of this manual.

6.1 SMC-351

The SMC-351 supports NAND flash memories.

This configuration supports a single memory interface, with the following configurable options:

- 32-bit or 64-bit AXI data width
- 8-bit or 16-bit memory data width
- 1-4 chip selects
- command FIFO depth
- read data FIFO depth
- write data FIFO depth
- *single-level cell* (SLC) *Error Correction Code* (ECC) block
- entry block pipeline.

See *Features of the SMC-35x series* on page 1-3 for a complete list of configuration options.

6.1.1 Functional overview

AXI reads are completed in the order they are accepted by the AXI interface, as are writes. The SMC-351 can prioritize between AXI reads and writes.

Only two clock domains exist:

- **aclk**
- **mclk0**.

6.1.2 Programmers model

This section describes the following registers:

- *direct_cmd*
- *set_opmode*
- *opmode*.

direct_cmd

Certain direct_cmd Register commands have no effect for this configuration because NAND memories do not require mode-register operations. The only cmd_type supported in the *Direct Command Register* on page 3-13 is b10.

set_opmode

In the *Set Operating Mode Register* on page 3-16, only the memory width field, set_mw, is relevant for NAND memories.

opmode

In the *Operating Mode Status Register* on page 3-21, only the memory width field, mw, is relevant for NAND memories.

6.1.3 Programmers model for test

This section describes the following registers:

- *int_inputs* on page 6-3
- *int_outputs* on page 6-3.

int_inputs

Only bits [5:0] that apply to the memory interface 0 are implemented.

int_outputs

Only bits [2:0] and bit [5] that apply to the memory interface 0 are implemented.

6.1.4 Signal descriptions

The pad interface only implements the NAND interface signals, see *NAND* on page B-12.

6.2 SMC-352

The SMC-352 supports a single SRAM memory interface type with the following configurable options:

- 32-bit or 64-bit AXI data width
- 8-bit, 16-bit, or 32-bit memory data width
- 1-4 chip selects
- command FIFO depth
- read data FIFO depth
- write data FIFO depth
- number of exclusive monitors
- entry block pipeline.

See *Features of the SMC-35x series* on page 1-3 for a complete list of configuration options.

6.2.1 Functional overview

Because the SMC-352 supports a single memory interface, AXI reads are completed in the order they are accepted by the AXI interface, as are writes. The SMC-352 can prioritize between AXI reads and writes.

Only two clock domains exist:

- **aclk**
- **mclk0**.

6.2.2 Programmers model

The SMC-352 implements the full functionality that Chapter 3 *Programmers Model* describes.

6.2.3 Programmers model for test

This section describes the following registers:

- *int_inputs*
- *int_outputs*.

int_inputs

Only bits [5:0] that apply to memory interface 0 are implemented.

int_outputs

Only bits [2:0] and bit [5] that apply to memory interface 0 are implemented.

6.2.4 Signal descriptions

The pad interface only implements the SRAM interface signals. See *SRAM* on page B-11.

6.3 SMC-353

The SMC-353 supports two memory interfaces:

Interface 0 type SRAM.

Interface 1 type NAND.

This configuration supports the following configurable options:

- 32-bit or 64-bit AXI data width
- 8-bit, 16-bit, or 32-bit memory data width for interface 0
- 8-bit, or 16-bit memory data width for interface 1
- 1-4 chip selects on each interface
- command FIFO depth
- read data FIFO depth
- write data FIFO depth
- number of exclusive monitors
- SLC ECC block for interface 1
- entry block pipeline.

See *Features of the SMC-35x series* on page 1-3 for a complete list of configuration options.

6.3.1 Functional overview

The SMC-353 implements the full functionality that Chapter 2 *Functional Description* describes.

6.3.2 Programmers model

Memory interface 0 supports the full functionality that Chapter 3 *Programmers Model* describes.

Memory interface 1 has programming restrictions for the following registers:

- *direct_cmd* on page 6-2
- *set_opmode* on page 6-2
- *opmode* on page 6-2.

direct_cmd

Certain direct_cmd Register commands have no effect for this configuration because NAND memories do not require mode-register operations. The only cmd_type supported in the *Direct Command Register* on page 3-13 is b10.

set_opmode

In the *Set Operating Mode Register* on page 3-16, only the memory width field, set_mw, is relevant for NAND memories.

opmode

In the *Operating Mode Status Register* on page 3-21, only the memory width field, mw, is relevant for NAND memories.

6.3.3 Programmers model for test

The SMC-353 implements the full functionality that Chapter 3 *Programmers Model* describes.

6.3.4 Signal descriptions

Interface 0 implements the SRAM interface signals, see *SRAM* on page B-11, with 0 appended, and Interface 1 implements the NAND interface signals, see *NAND* on page B-12, with 1 appended.

6.4 SMC-354

The SMC-354 supports two memory interfaces, both are of type SRAM.

This configuration supports the following configurable options:

- 32-bit or 64-bit AXI data width
- 8-bit, 16-bit, or 32-bit memory data width for interface 0
- 8-bit, 16-bit, or 32-bit memory data width for interface 1
- 1-4 chip selects on each interface
- command FIFO depth
- read data FIFO depth
- write data FIFO depth
- number of exclusive monitors
- entry block pipeline.

See *Features of the SMC-35x series* on page 1-3 for a complete list of configuration options.

6.4.1 Functional overview

The SMC-354 implements the full functionality that Chapter 2 *Functional Description* describes.

6.4.2 Programmers model

The SMC-354 implements the full functionality that Chapter 3 *Programmers Model* describes.

6.4.3 Programmers model for test

The SMC-354 implements the full functionality that Chapter 4 *Programmers Model for Test* describes.

6.4.4 Signal descriptions

Both memory interfaces implement the SRAM interface signals. See *SRAM* on page B-11.

Interface 0 signals have a 0 appended and interface 1 signals have a 1 appended.

Appendix A
Revisions

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

Table A-1 Differences between issue G and issue H

Change	Location	Affects
All NAND control and data outputs are registered on the rising edge of **mclkn**, which is equivalent to the *falling* edge of **mclk**.	*NAND interface timing diagrams* on page 2-37	r2p2
set_cycles register bit assignments, bits[13:11], bits[10:8].	Table 3-7 on page 3-15	r2p2
State information for the ecc_fail field in the ECC Status Register has been added.	Table 3-16 on page 3-25	r2p2
Bits[31:30] of ecc<x>_status are reserved.	Table 3-16 on page 3-25	r2p2
ECC block mode b01 in ecc<x>_cfg[3:2], ECC values are not read from or written to memory.	Table 3-17 on page 3-27	r2p2
When ecc<x>_cfg[1:0]==b00, this field is still reserved if ecc_extra_block is configured but not enabled.	Table 3-17 on page 3-27	r2p2
Turnaround time t_{TR}, is enforced between any two consecutive accesses in multiplexed address/data mode.	Figure 2-24 on page 2-35	r2p2

Table A-1 Differences between issue G and issue H (continued)

Change	Location	Affects
In synchronous burst read in multiplexed mode, the address is held on the data bus while **adv** is low and for the duration of the next cycle.	Figure 2-21 on page 2-32	r2p2
In synchronous burst write in multiplexed mode, the address is held on the data bus while **adv** is low and for the duration of the next cycle.	Figure 2-23 on page 2-34	r2p2
opmode register bits [15:13] reset to b000.	Table 3-13 on page 3-22	r2p2
Opmode register bits[31:16] and [1:0] are dependent on external tie-offs.	Table 3-1 on page 3-5, see footnote	r2p2
periph_id_3 bit[0] is reserved.	Table 3-28 on page 3-34 Figure 3-28 on page 3-32	r2p2
The SMC detects but does not correct errors.	*Features of the SMC-35x series* on page 1-3 Table 3-16 on page 3-25, bits[24:20], see footnote *Correcting errors* on page 2-48 *Integration Outputs Register* on page 4-4, usage constraints	r2p2
ecc<x>_memcmd0 and ecc<x>_memcmd1 registers.	Table 3-18 on page 3-28 Table 3-19 on page 3-29	r2p2
SRAM memory widths and AXI data widths.	Table 1-1 on page 1-4 and Table 1-2 on page 1-4 Table 1-2 on page 1-4	r2p2
Updated register map to show all of the ECC registers.	Figure 3-4 on page 3-4	r2p0
Updated access type from RO to RW for: • ecc_status Register • ecc<n>_block[3:0] Register.	Table 3-1 on page 3-5	r2p0
ecc_int1_en bit changed to ecc_int_en1. ecc_int0_en bit changed to ecc_int_en0.	*Memory Controller Status Register* on page 3-8	r2p0
Register name changed from memc_cfg_set Register to mem_cfg_set Register.	*Set Configuration Register* on page 3-11	r2p0
Register name changed from memc_cfg_clr Register to mem_cfg_clr Register. Bit [6] changed from ecc_int_disable0 to ecc_int_disable1. int_clr_1 bit changed to int_clear1. int_clr_0 bit changed to int_clear0.	*Clear Configuration Register* on page 3-12	r2p0
set_bls bit changed to set_bls_time.	*Set Operating Mode Register* on page 3-16	All revisions
Register name changed from refresh_period_0 Register to refresh_0 Register. period field changed to ref_period0.	*Refresh Period 0 Register* on page 3-18	r2p0

Table A-1 Differences between issue G and issue H (continued)

Change	Location	Affects
Register name changed from refresh_period_1 Register to refresh_1 Register. period field changed to ref_period1.	*Refresh Period 1 Register* on page 3-19	r2p0
Updated the reset value of the burst_align<x> field from b001 to b000. address_match<x> field changed to add_match<x>. address_mask<x> field changed to add_mask<x>. bls<x> bit changed to bls_time<x>.	*Operating Mode Status Register* on page 3-21	All revisions
Added requirement to set **csysack** HIGH when the controller exits integration test mode.	*Integration Configuration Register* on page 4-3	All revisions
Added an _int suffix to all of the bit names.	*Integration Outputs Register* on page 4-4	All revisions
Updated signal type for **add_<x>[31:0]**.	Table B-16 on page B-10	All revisions

Appendix B
Signal Descriptions

This appendix describes the signals that the SMC uses. It contains the following sections:

B.1 Clock and reset signals

Table B-1 shows the AXI and APB clock and reset signals.

Table B-1 Clock and reset signals

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

Table B-2 shows the memory interface 0 clock and reset signals.

Table B-2 Memory interface 0 clock and reset signals

Signal	Type	Source	Description
mclk0	Input	Clock source	Memory interface 0 clock.
mclk0n	Input	Clock source	Memory interface 0 clock inverted.
mreset0n	Input	Reset source	Reset for **mclk0** domain. This signal is active LOW.

Table B-3 shows the memory interface 1 clock and reset signals.

Table B-3 Memory interface 1 clock and reset signals

Signal	Type	Source	Description
mclk1	Input	Clock source	Memory interface 1 clock.
mclk1n	Input	Clock source	Memory interface 1 clock inverted.
mreset1n	Input	Reset source	Reset for **mclk1** domain.

B.2 Miscellaneous signals

The following sections describe the miscellaneous signals:

- *SRAM miscellaneous signals*
- *NAND miscellaneous signals* on page B-4
- *Interrupt signals* on page B-4
- *Tie-off signals* on page B-4
- *User signals* on page B-5
- *Scan test* on page B-5.

B.2.1 SRAM miscellaneous signals

Table B-4 shows the SRAM miscellaneous signals.

Table B-4 SRAM miscellaneous signals

Signal[a]	Type	Source	Description
mux_mode_<x>	Input	Tie-off	When HIGH, the memory interface operates in multiplexed address/data mode.
remap_<x>	Input	Tie-off	When HIGH, the SMC remaps chip select 0, on memory interface <x>, to address `0x0`.
sram_mw_<x>[1:0]	Input	Tie-off	Sets the memory width for chip select 0, on memory interface <x>, when **remap_<x>** is HIGH. The encoding is: b00 = 8-bit b01 = 16-bit b10 = 32-bit b11 = Reserved.

a. The **<x>** notation represents memory interface 0 or 1.

B.2.2 NAND miscellaneous signals

Table B-5 shows the NAND miscellaneous signals.

Table B-5 NAND miscellaneous signals

Signal[a]	Type	Source	Description
nand_booten_<x>	Input	Tie-off	Enable NAND booting functionality.
nand_csl_<x>	Input	Tie-off	When HIGH, the chip select remains asserted between the address phase and data phase of a transfer on the NAND interface.
nand_mw_<x>	Input	Tie-off	Sets the memory width for chip select 0, on memory interface <x>, when **remap_<x>** is HIGH: 0 = 8-bit 1 = 16-bit.
remap_<x>	Input	Tie-off	When HIGH, the SMC remaps chip select 0, on memory interface <x>, to address `0x0`.

a. The **<x>** notation represents memory interface 0 or 1.

B.2.3 Interrupt signals

Table B-6 shows the interrupt signals.

Table B-6 Interrupt signals

Signal	Type	Destination	Description
smc_int	Output	Interrupt controller	Combined interrupt output
smc_int<x>[a]	Output	Interrupt controller	Individual memory interrupt outputs
ecc_int<x>[b]	Output	Interrupt controller	Individual ECC interrupt outputs

a. The **<x>** notation represents memory interface 0 or 1. For single memory interface configurations without ECC, only the **smc_int** signal is present.

b. **ecc_int<x>** is only present for NAND interfaces with ECC configured

B.2.4 Tie-off signals

Table B-7 shows the tie-off signals.

Table B-7 Tie-off signals

Signal	Type	Source	Description
addr_mask<x>_<n>[7:0][a]	Input	Tie-off	Address mask for chip select <n>. A mask applied to the AXI address bits [31:24] before the comparison with the address_match value.
addr_match<x>_<n>[7:0][a]	Input	Tie-off	Address match for chip select <n>. The comparison value that determines the chip select base address.
async0	Input	Tie-off	When HIGH, indicates to **aclk** domain that **aclk** is synchronous to **mclk0**. When LOW, synchronizing logic is enabled.

Table B-7 Tie-off signals (continued)

Signal	Type	Source	Description
async1	Input	Tie-off	When HIGH, indicates to **aclk** domain that **aclk** is synchronous to **mclk1**. When LOW, synchronizing logic is enabled.
a_gt_m0_sync	Input	Tie-off	Set this signal HIGH if **aclk** is greater than **mclk0** but is still synchronous.
a_gt_m1_sync	Input	Tie-off	Set this signal HIGH if **aclk** is greater than **mclk1** but is still synchronous.
msync0	Input	Tie-off	When HIGH, indicates to **mclk0** domain that **mclk0** is synchronous to **aclk**. When LOW, synchronizing logic is enabled.
msync1	Input	Tie-off	When HIGH, indicates to **mclk1** domain that **mclk1** is synchronous to **aclk**. When LOW, synchronizing logic is enabled.
use_ebi	Input	Tie-off	Set this signal HIGH if a memory interface of the SMC connects to a PrimeCell External Bus Interface (PL220).

a. The **<x>** notation represents memory interface 0 or 1. The **<n>** notation indicates that you can use chip select 0 to 3.

B.2.5 User signals

These are general purpose I/O signals that you can use to control and monitor external devices. Table B-8 shows the user signals.

Table B-8 User signals

Signal	Type	Source or destination	Description
user_config[7:0]	Output	External control logic	General purpose output signals that you program using the *User Config Register* on page 3-24
user_status[7:0]	Input	External control logic	General purpose input signals that are read using the *User Status Register* on page 3-23

B.2.6 Scan test

Table B-9 shows the scan test signals.

Table B-9 Scan test signals

Signal	Type	Source	Description
dft_en_clk_out	Input	Tie-off	Used to force the **clk_out[ ]** outputs to be the same as **mclk<x>**. This signal is used for ATPG testing.
rst_bypass	Input	Tie-off	Used to bypass synchronization of external resets. This signal is used for ATPG testing.

B.3 AXI interface signals

The following sections describe the AXI signals:

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

B.3.1 Write address channel signals

Table B-10 shows the AXI write address channel signals.

Table B-10 Write address channel signals

Signal	AMBA equivalent[a]
awaddr[31:0]	**AWADDR**
awburst[1:0]	**AWBURST[1:0]**
awcache[3:0][b]	**AWCACHE[3:0]**
awid[7:0]	**AWID**
awlen[3:0]	**AWLEN[3:0]**
awlock[1:0]	**AWLOCK[1:0]**
awprot[2:0][b]	**AWPROT[2:0]**
awready	**AWREADY**
awsize[2:0]	**AWSIZE[2:0]**
awvalid	**AWVALID**

a. See the *AMBA AXI Protocol v2.0 Specification* for a description of these signals.

b. The SMC ignores any information that it receives on these signals.

B.3.2 Write data channel signals

Table B-11 shows the AXI write data channel signals.

Table B-11 Write data channel signals

Signal	AMBA equivalent[a]
wdata[PORTWIDTH–1:0][b]	**WDATA**
wid[7:0]	**WID**
wlast	**WLAST**

Table B-11 Write data channel signals (continued)

Signal	AMBA equivalent[a]
wready	**WREADY**
wstrb[PORTBYTES–1:0][b]	**WSTRB**
wvalid	**WVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.

b. The value of **PORTWIDTH** is set during configuration of the SMC. **PORTBYTES**=**PORTWIDTH**÷8.

B.3.3 Write response channel signals

Table B-12 shows the AXI write response channel signals.

Table B-12 Write response channel signals

Signal	AMBA equivalent[a]
bid[7:0]	**BID**
bready	**BREADY**
bresp[1:0][b]	**BRESP[1:0]**
bvalid	**BVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.

b. The SMC ties **bresp[1]** LOW and therefore it only provides OKAY or EXOKAY responses.

B.3.4 Read address channel signals

Table B-13 shows the AXI read address channel signals.

Table B-13 Read address channel signals

Signal	AMBA equivalent[a]
araddr[31:0]	**ARADDR**
arburst[1:0]	**ARBURST[1:0]**
arcache[3:0][b]	**ARCACHE[3:0]**
arid[7:0]	**ARID**
arlen[3:0]	**ARLEN[3:0]**
arlock[1:0]	**ARLOCK[1:0]**
arprot[2:0][b]	**ARPROT[2:0]**
arready	**ARREADY**
arsize[2:0]	**ARSIZE[2:0]**
arvalid	**ARVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The SMC ignores any information that it receives on these signals.

B.3.5 Read data channel signals

Table B-14 shows the AXI read data channel signals.

Table B-14 Read data channel signals

Signal	AMBA equivalent[a]
rdata[PORTWIDTH–1:0][b]	**RDATA**
rid[7:0]	**RID**
rlast	**RLAST**
rready	**RREADY**
rresp[1:0][c]	**RRESP[1:0]**
rvalid	**RVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The value of **PORTWIDTH** is set during configuration of the SMC.
c. The SMC ties **rresp[1]** LOW and therefore it only provides OKAY or EXOKAY responses.

B.3.6 AXI low-power interface signals

Table B-15 shows the AXI low-power interface signals.

Table B-15 AXI low-power interface signals

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

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.

B.4 APB signals

Table B-16 shows the APB signals.

Table B-16 APB interface signals

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

a. See the *AMBA 3 APB Protocol Specification* for a description of these signals.
b. The SMC only uses **paddr[11:2]** and ignores the other address bits.
c. The SMC ties this signal LOW.

B.5 Pad interface signals

This following section describes the SRAM and NAND pad interface signals:

- *SRAM*
- *NAND* on page B-12.

B.5.1 SRAM

Table B-17 shows the SRAM pad interface signals.

Table B-17 SRAM pad interface signals

Signal[a]	Type	Source or destination	Description
add_<x>[31:0]	Output	External memory	Address
adv_n_<x>	Output	External memory	Address valid
baa_n_<x>	Output	External memory	Burst address advance
bls_n_<x>[(MEMWIDTH÷8)–1:0][b]	Output	External memory	Byte lane strobe
clk_out_<x>[MEMORIES–1:0][c]	Output	External memory	Memory clock
cre_<x>	Output	External memory	Configuration register write
cs_n_<x>[MEMORIES–1:0][b]	Output	External memory	Chip select
data_en_<x>	Output	External memory	Data bus tristate enable
data_in_<x>[MEMWIDTH–1:0][a]	Input	External memory	Data in
data_out_<x>[MEMWIDTH–1:0][a]	Output	External memory	Data out
fbclk_in_<x>	Input	External memory	Feedback clock
int_<x>	Input	External memory	Interrupt
oe_n_<x>	Output	External memory	Output enable
wait_<x>	Input	External memory	Wait
we_n_<x>	Output	External memory	Write enable

a. The **<x>** notation represents memory interface 0 or 1.

b. MEMWIDTH is the data width of the external memory bus in bits and is set during configuration of the SMC.

c. MEMORIES is the number of chip selects and is set during configuration of the SMC.

B.5.2 NAND

Table B-18 shows the NAND pad interface signals.

Table B-18 NAND pad interface signals

Signal[a]	Type	Source or destination	Description
ale_<x>	Output	External memory	Address latch enable
busy_<x>	Input	External memory	Busy
cle_<x>	Output	External memory	Command latch enable
cs_n_<x>[MEMORIES–1:0][b]	Output	External memory	Chip select
data_en_<x>	Output	External memory	Data bus tristate enable
data_in_<x>[MEMWIDTH–1:0][c]	Input	External memory	Data in
data_out_<x>[MEMWIDTH–1:0][c]	Output	External memory	Address and data out
re_n_<x>	Output	External memory	Read enable
we_n_<x>	Output	External memory	Write enable

a. The **<x>** notation represents memory interface 0 or 1.

b. MEMORIES is the number of chip selects and is set during configuration of the SMC.

c. MEMWIDTH is the data width of the external memory bus in bits and is set during configuration of the SMC.

B.6 EBI signals

Table B-19 shows the *External Bus Interface* (EBI) signals.

Table B-19 EBI signals

Signal[a]	Type	Source or destination	Description
ebibackoff<x>	Input	SMC	External memory bus access backoff. The EBI backoff signal goes HIGH when the EBI wants to remove the SMC from the memory bus so that another memory controller can be granted the memory bus.
ebigrant<x>	Input	SMC	External memory bus grant. This signal goes HIGH when the EBI grants the external memory bus to the SMC.
ebireq<x>	Output	External	External memory bus request. The SMC sets this signal HIGH when it requires the memory bus.

a. The **<x>** notation represents memory interface 0 or 1.